150 ptas.

mi computer

22

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

9 788485 822836 00022

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen II - Fascículo 22

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-90-0 (tomo 2)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 138406
Impreso en España - Printed in Spain - Junio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Diseños de calidad

El diseño auxiliado por ordenador pide difíciles cálculos para sus refinados productos. Muchos de sus principios son aplicables por ordenadores personales



Cortesía de Soft

Ian McKinnell

La flor del manzano
Por muy sofisticado que sea el software, el elemento más importante de un paquete de diseño auxiliado por ordenador es el diseñador que lo utiliza. Versawriter, cuyos resultados vemos en la fotografía, funciona en un Apple II y exige dos unidades de disco y un digitalizador. Un usuario experimentado tardó cierto tiempo en dar entrada a la flor en la máquina

La idea de aplicar el ordenador al proceso del diseño industrial surgió por primera vez en la década de los sesenta (en el Massachusetts Institute of Technology). No obstante, habría de transcurrir una década más hasta que la tecnología del ordenador le permitiera al diseñador ver una representación gráfica de su trabajo e interactuar con ella. Una vez presente en la pantalla de un monitor, se accede a su modificación mediante un digitalizador o un lápiz óptico, exactamente igual que si se estuviera frente a un tablero de dibujo. Estos periféricos esenciales (el digitalizador, el lápiz óptico y el plotter) son las herramientas básicas del arte del diseño auxiliado por ordenador. Con ellos se pueden crear imágenes de la misma forma como las crea un animador (véase p. 181), "dibujando" en una tablilla de digitalización. El diseñador puede modificar dicha imagen, tal vez utilizando un lápiz óptico, incorporando componentes predibujados y subensamblajes, y después sacar una copia del dibujo acabado con un plotter. El ordenador se convierte en un sistema de delineación en principio similar a un procesador de textos, pero trabajando con imágenes en lugar de letras.

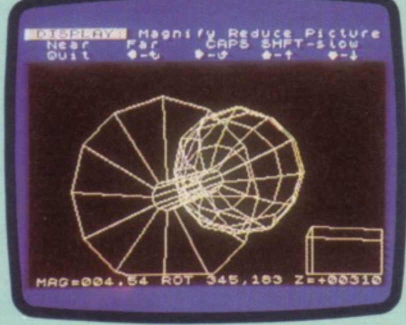
Hemos visto que la calidad de la imagen producida en el monitor de un ordenador depende de dos cosas: de la capacidad de resolución del monitor (es decir, las dimensiones de un elemento o pixel individual de la imagen) y de la potencia y las dimensiones de la memoria del ordenador que acciona el monitor. Cuando analizamos las imágenes generadas por ordenador, nos encontramos en gran medida con las mismas exigencias: un monitor que pudiera resolver algo así como 1 000 × 1 200 pixels, y un ordenador capaz de procesar estos 1,2 millones de elementos de la imagen en menos de 1/24 segundos.

Nos será útil continuar con la analogía entre el paquete para diseño auxiliado por ordenador y el procesador de textos. En lugar de trasladar párrafos, oraciones o palabras individuales a través de un trozo de texto (corrigiendo, insertando o borrándolos a voluntad) el programa CAD (*Computer Assisted Design*: diseño auxiliado por ordenador) se puede emplear para trasladar a través de la página los elementos de un dibujo. El efecto podrá ser diferente, pero el principio es exactamente el mismo.

Desde el punto de vista del programa, se trata de cómo almacenar esta imagen de forma tal que se la pueda alterar y manipular. Si fuéramos a realizar el modelo físico de un objeto, utilizaríamos uno de dos métodos básicos: aditivo o sustractivo. El método aditivo es como modelar en arcilla: se va construyendo el objeto pieza tras pieza hasta que llegamos a la forma final. El método sustractivo se parece al que siguen los escultores, que sacan la obra





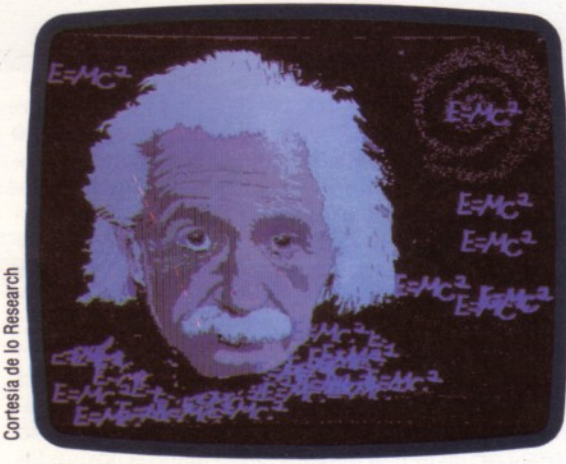




Un toque profesional
No todo el software de gráficos y de CAD es caro. VU-3D de Psion, para el Sinclair Spectrum de 48 Kbytes, ofrece la mayoría de las posibilidades de los paquetes profesionales (aunque, por supuesto, a un nivel mucho menos refinado) y cuesta muy poco

Ian McKinnell

desbrozando la materia. Por analogía, en el ordenador el bloque de materia sólida es una matriz tridimensional. Por consiguiente, adquieren importancia las dimensiones y el rendimiento del ordenador utilizado. Si la matriz es lo suficientemente grande como para permitir que se destine todo un byte para la definición de cada pixel o elemento de la imagen, entonces la cantidad de información que podremos retener acerca de cada elemento es bastante grande (256 partes separadas al emplear un procesador de ocho bits, y muchas más todavía para los dispositivos de 16 o 32 bits). Pero los problemas que plantea la creación de tanto espacio de almacenamiento son prácticamente insolubles, de modo que nos vemos obligados a asumir un compromiso, que por lo general es bastante aceptable. En lugar de destinar un byte entero para cada elemento, es suficiente destinar un único bit, si todo lo que deseamos hacer es indicar la presencia o la ausencia de un elemento en esta posición del modelo.



Fórmula eficiente
El sistema Pluto, de Io Research, lleva la generación de imágenes en alta resolución a una amplia gama de pequeños microordenadores, añadiendo, además, un procesador veloz y memoria extra. El sistema básico no es excesivamente caro y proporciona ocho colores fijos y un poder de resolución de 670 × 576 pixels

Cortesía de Io Research

El software para diseño auxiliado por ordenador comparte muchas de las características de los paquetes de imágenes generadas por ordenador: atenuación de curvas, eliminación de elementos ocultos, sombreado, relleno y recoloreado de zonas, por ejemplo. Para formar una curva sólo se requiere la solución repetida de una ecuación simple para una serie de valores. Especificando los puntos de comienzo y final para una línea determinada, y la distancia máxima que alcanzará la curva respecto a dicha línea recta, proporcionamos una solución para la ecuación. A partir de esta solución podemos trabajar a la inversa, para deducir la ecuación en sí misma, y después proceder a resolverla para el resto de la serie de valores, formando así la curva.

Esta capacidad para componer un dibujo a partir de partes estándar de los componentes es el verdadero punto fuerte de los sistemas CAD. Ya no es necesario volver a dibujar los componentes individuales comunes. Una vez que éstos se han defini-

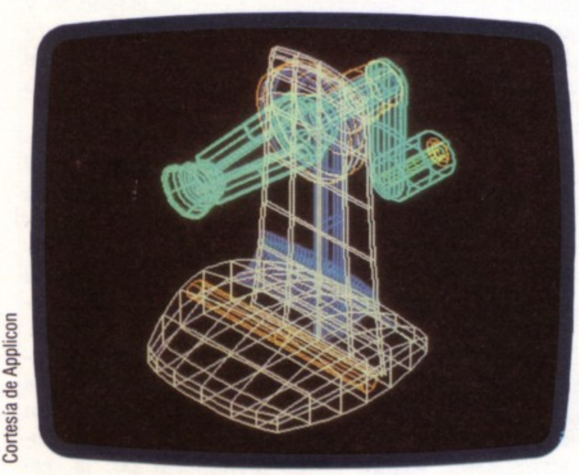


do, se puede volver a llamar dicha definición cada vez que se la requiera e incorporarla a nuevos dibujos. Un ejemplo destacable de esto es el uso de ordenadores para auxiliar el diseño de las futuras generaciones de ordenadores.

El diseño de un circuito impresor, por ejemplo, es muy complicado, por lo que requiere la aplicación de técnicas de optimización para acomodar los componentes y sus pasos de interconexión de la forma más económica posible (teniendo en cuenta que las vías de conexión nunca se han de cruzar entre sí). El diseñador a menudo debe remitirse al ensayo y error, y es aquí donde los paquetes CAD son especialmente útiles. Todos los componentes individuales se almacenan como imágenes predefinidas y se les llama cuando se necesitan. Resulta muy sencillo probar un diseño determinado en la unidad de visualización de datos, para ver si responde a todas las exigencias, antes de trasladarlo al papel como parte de un dibujo en ejecución. Siguiendo este método se puede esbozar un diseño e incluso probar diferentes soluciones, empleando el tiempo que llevaría completar un boceto.

Los circuitos integrados se diseñan casi exactamente de la misma manera, pero debido a la densidad de componentes y vías de conexión, se requiere además otra configuración de software: la capacidad para ampliar una parte del dibujo, trabajar en él a escala aumentada y luego devolverlo a su posición dentro del diseño global. Este efecto es hoy una pieza básica del repertorio CAD y ha representado una considerable mejora en cuanto a la eficacia del sistema. Mediante su utilización se puede retener la especificación de un objeto completo en un solo dibujo, y se puede ajustar la escala para satisfacer las necesidades de quien lo mira.

Y no es sólo la escala lo único que se puede variar de este modo. Si consideramos el caso de un objeto más complejo (un coche, p. ej.), estamos ante múltiples subsistemas que, juntos, conforman el todo: el sistema eléctrico, el sistema hidráulico, el de escape, la suspensión, etc. Mientras que al diseñador de estética le interesará más el paquete global, a los ingenieros individuales probablemente lo que más les interese sea un solo subsistema. Es muy sencillo hacer que cada subsistema vaya en un color distinto para extraer después del dibujo global, todos los objetos de un color determinado. Esto no equivale a afirmar que el dibujo siempre ha de estar constituido por una mezcla de colores; la codificación se puede suprimir a voluntad cuando no se la considere necesaria.



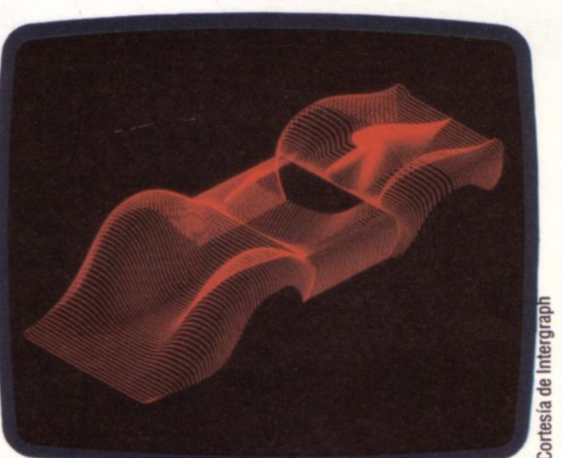
Cortesía de Applicon



Cortesía de Siggraph

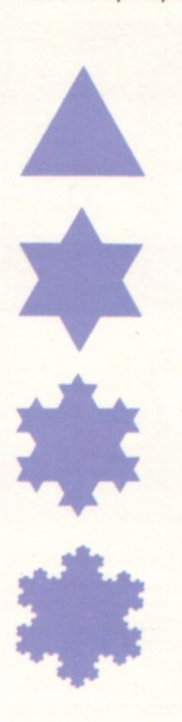
El avance más sensacional es la capacidad para retener la especificación *completa* de un objeto (no sólo su forma y su aspecto, sino también información relativa al material en el que está construido, su peso, costo, etc.). Recabar información acerca de la forma y el tamaño del objeto es sólo una función del sistema, que se puede considerar como una base de datos orientada hacia la visualización. Formulando distintas preguntas a dicha base de datos se pueden ordenar pedidos a los proveedores, planificar la fabricación de subensamblajes y componentes, integrar las cadenas de producción para asegurar que los componentes lleguen exactamente a donde y cuando se les necesita, analizar costos y controlar la eficacia de la fabricación, entre otras muchas cosas. Estamos tentados de imaginar que el paso siguiente será un sistema regular de control directo de la fabricación por ordenador.

La mayoría de las aplicaciones que hemos analizado aquí requieren ordenadores de unidad principal o bien miniordenadores muy potentes, pero ello no quiere decir que incluso los microordenadores pequeños no puedan desempeñar un papel útil en el proceso de diseño. Existe una amplia gama de software para CAD destinado a máquinas que trabajan bajo CP/M, por ejemplo, y la mayoría de los fabricantes ofrecen al menos un paquete, incluso para ordenadores tan poco sofisticados, relativamente, como el Sinclair ZX81. Como hemos apuntado, el tamaño y la velocidad del ordenador determinan la calidad de la imagen almacenada, pero es poco probable que las necesidades del usuario de un ordenador personal sean las de un diseñador profesional, de modo que es factible conseguir brillantes resultados por un desembolso modesto.



Cortesía de Intergraph

Técnica e imaginación
El fondo de esta vista de la producción de Lucasfilms *Road to Point Reyes* se compuso en gran medida mediante "fractals", una nueva e ingeniosa técnica de CAD. Los "fractals" son fenómenos cuya complejidad aumenta según vamos contemplándolos más cerca. Las colinas y las montañas que aparecen al fondo se crearon como polígonos simples, descritos dentro de la memoria de un ordenador. Luego, cada polígono se fue haciendo progresivamente más complejo mediante la adición de su propia forma a cada uno de sus lados, repitiéndose el proceso con un cierto grado de aleatoriedad. El desarrollo de la forma de un copo de nieve, que vemos abajo, a partir de un simple triángulo, sirve para ilustrar este principio



Estructura básica
La primera etapa en la creación de una imagen o diseño tridimensional se conoce como "estructuración lineal". La imagen se define como una serie de coordenadas de puntos, unidas adecuadamente mediante líneas rectas. Estas líneas se pueden manipular utilizando el algoritmo de suavización de curvas, eliminando las líneas ocultas y después rellenando con color y sombreando, según sea necesario, para aumentar la ilusión de profundidad

La cinta teórica

La máquina Turing es un dispositivo puramente teórico, que se utiliza para decidir si un problema es informatizable o no

En *Mi Computer* solemos seleccionar temas prácticos y cosas que usted puede hacer con su ordenador personal. Sin embargo, en este apartado vamos a echar una mirada al lado teórico de los ordenadores: al campo que se denomina *ciencia de los ordenadores*. Tal ciencia es a la informática lo que las matemáticas puras son a la ingeniería, es decir, un campo que se caracteriza por ser extremadamente teórico pero del cual derivan las ideas prácticas.

La máquina Turing, por ejemplo, es una idea puramente teórica que desarrolló Alan Turing (véase p. 200) como ayuda para estudiar los algoritmos y la posibilidad de informatización. Se trata en realidad del "mínimo ordenador posible", de modo que si se puede demostrar que un problema determinado no se puede resolver utilizando una máquina Turing, entonces se puede afirmar que dicho problema no es "informatizable". Turing pensó que un ordenador mínimo de este tipo necesitaría tres configuraciones: un almacenamiento externo para grabar y almacenar la información de entrada y de salida; un medio para leer dicho almacenamiento y para escribir en él, y en tercer lugar una unidad de control que permitiese determinar las acciones a emprender.

Por definición, una máquina Turing posee, entonces, una cinta (si le sirve de ayuda, imagínesela como una cinta magnética) de longitud infinita (es decir: sea cual fuere la cantidad de cinta necesaria para solucionar un problema, siempre habrá la suficiente). La cinta está dividida en cuadrados, que o bien estarán en blanco o contendrán un símbolo. A lo largo de la cinta se mueve una "cabeza", que puede leer o escribir los símbolos en los cuadrados y que recibe sus instrucciones desde una unidad de control que le proporciona una doble indicación: cuáles son los símbolos que debe escribir y en qué dirección se ha de mover a continuación.

La unidad de control posee un programa de ejecución, y en este sentido se puede decir que cada máquina Turing se "fabrica" específicamente para realizar una aplicación, ya que en la especificación no existen medios para cargar o alterar un programa. Hemos entrecomillado la palabra fabricar porque las únicas máquinas Turing que se han construido físicamente lo fueron con fines puramente educativos. Sin embargo, escribir un programa en BASIC que imite el funcionamiento de una máquina Turing en un ordenador personal es un ejercicio relativamente sencillo.

El programa de control de una máquina Turing consta de un conjunto de "quíntuplos", o sentencias que contienen cinco elementos. Qué quíntuplo se ejecuta en cada etapa depende de dos factores: el símbolo que contenga el cuadrado que esté debajo de la cabeza de cinta, y el "estado" o "condición" de la máquina. Dicho estado es una cualidad estrictamente arbitraria: podemos especificar que la máquina se ponga en marcha en el estado SA y que cuando llegue al estado especial H entonces se detenga, dándose por acabado el cálculo. Entretanto, el estado cambiará muchas veces de acuerdo con las instrucciones de los "quíntuplos". El estado tan sólo refleja lo que ha sucedido hasta el momento en el cálculo y ayuda a seleccionar qué "quíntuplo" se ejecutará a continuación (nuevamente, si le sirve de ayuda, imagíneselo como una variable de bandera en programación BASIC).

Los cinco elementos de que consta cada quíntuplo son:

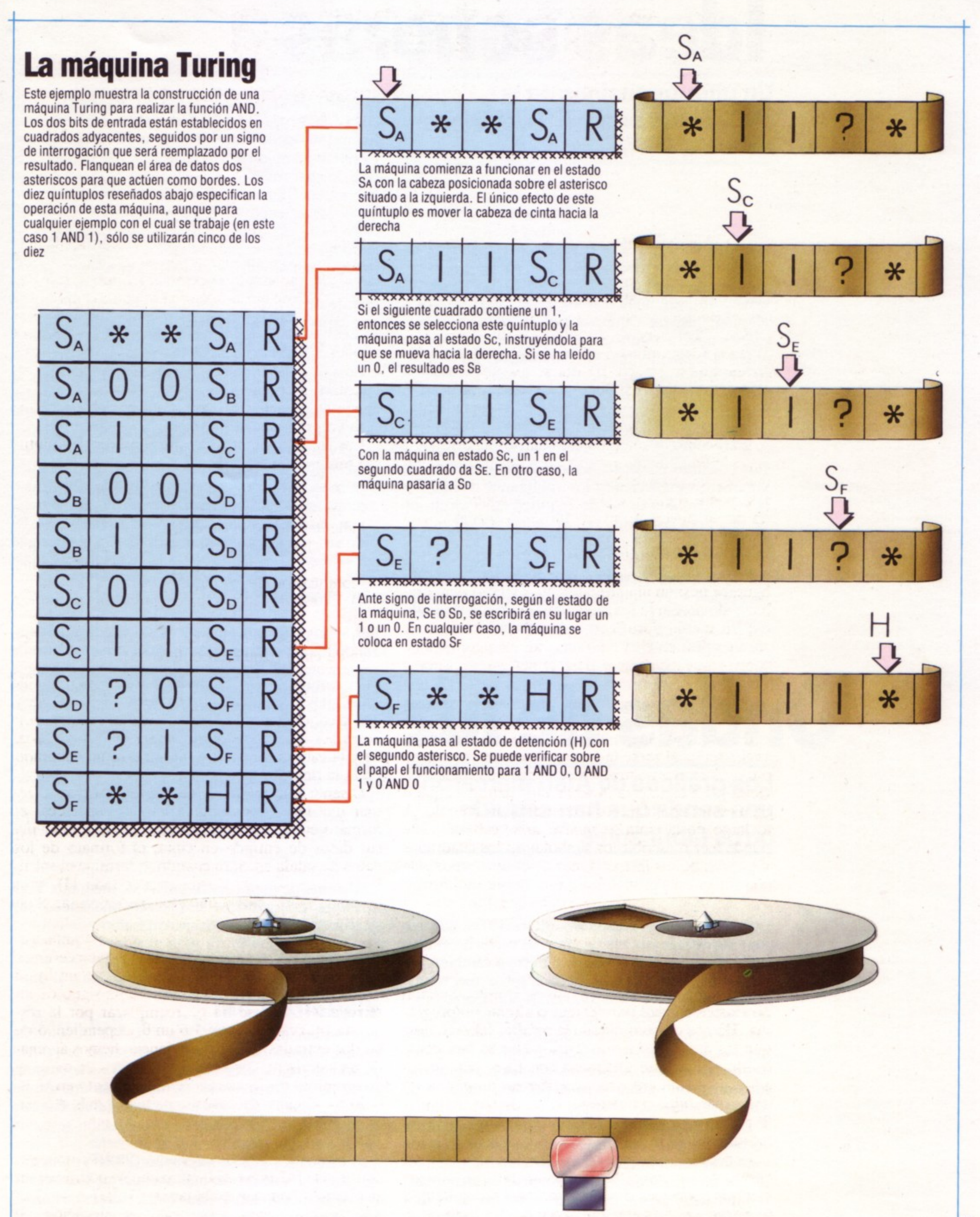
1) El estado en curso de la máquina;
2) El símbolo del cuadrado de la cinta que se halle debajo de la cabeza;
3) El símbolo a escribir en ese cuadrado, que será el mismo que en 2) si no se requiere ninguna modificación de los datos;
4) El estado en que la máquina ha de entrar ahora;
5) La dirección en la que se debe mover la cabeza de cinta (izquierda o derecha).

El "quíntuplo" (SA,5,3,SB,D), por ejemplo, se ejecutará siempre que la máquina esté en estado SA y que la cabeza de cinta lea un 5. El 5 se reemplazará luego por un 3, la máquina pasará del estado SA al SB y la cabeza de cinta se desplazará un cuadrado hacia la derecha.

Diseñar una máquina Turing teórica para efectuar una tarea determinada implica especificar el formato en el cual se le presentarán a la máquina sus datos de entrada en cinta, el formato de los datos de salida en cinta cuando se termine el cálculo (es decir, con la máquina en estado H), y el grupo de quíntuplos requeridos para ejecutar el algoritmo.

En el recuadro hemos diseñado una máquina Turing para realizar la función AND. Estableceremos los dos bits de entrada (cada uno un 1 o un 0) en cuadrados adyacentes, seguidos de un signo de interrogación, que se ha de reemplazar por la respuesta (nuevamente, un 1 o un 0, dependiendo de las dos entradas). Por conveniencia, hemos agregado un asterisco a los extremos del área de datos, y pondremos la máquina en funcionamiento en el estado SA encima del asterisco situado más a la izquierda, acabando sobre el situado más a la derecha.

Se necesita un total de diez quíntuplos para especificar esta máquina, si bien, como se podrá ver en el ejemplo con que trabajamos (1 AND 1 = 1) se utilizan sólo cinco para cualquier ejecución. Si usted prueba la misma máquina para, supongamos, 0 AND 1, descubrirá que de los diez quíntuplos se selecciona un grupo de quíntuplos diferente.



La máquina Turing

Este ejemplo muestra la construcción de una máquina Turing para realizar la función AND. Los dos bits de entrada están establecidos en cuadrados adyacentes, seguidos por un signo de interrogación que será reemplazado por el resultado. Flanquean el área de datos dos asteriscos para que actúen como bordes. Los diez quíntuplos reseñados abajo especifican la operación de esta máquina, aunque para cualquier ejemplo con el cual se trabaje (en este caso 1 AND 1), sólo se utilizarán cinco de los diez

S_A	*	*	S_A	R
S_A	0	0	S_B	R
S_A	1	1	S_C	R
S_B	0	0	S_D	R
S_B	1	1	S_D	R
S_C	0	0	S_D	R
S_C	1	1	S_E	R
S_D	?	0	S_F	R
S_E	?	1	S_F	R
S_F	*	*	H	R

La máquina comienza a funcionar en el estado S_A con la cabeza posicionada sobre el asterisco situado a la izquierda. El único efecto de este quíntuplo es mover la cabeza de cinta hacia la derecha

Si el siguiente cuadrado contiene un 1, entonces se selecciona este quíntuplo y la máquina pasa al estado S_C, instruyéndola para que se mueva hacia la derecha. Si se ha leído un 0, el resultado es S_B

Con la máquina en estado S_C, un 1 en el segundo cuadrado da S_E. En otro caso, la máquina pasaría a S_D

Ante signo de interrogación, según el estado de la máquina, S_E o S_D, se escribirá en su lugar un 1 o un 0. En cualquier caso, la máquina se coloca en estado S_F

La máquina pasa al estado de detención (H) con el segundo asterisco. Se puede verificar sobre el papel el funcionamiento para 1 AND 0, 0 AND 1 y 0 AND 0

Ideas sonoras

Un ulterior análisis de la sofisticada orden ENVELOPE del BBC Micro

En el capítulo anterior de *Sonido y luz* presentamos la orden ENVELOPE del BBC Micro. Es una de las órdenes más poderosas de que dispone el programador de BASIC, cuando se utiliza junto con la orden SOUND, analizada en la p. 358. Vamos a ahondar nuestra explicación de ENVELOPE viendo qué es una *envoltura de volumen*.

En la siguiente línea de parámetros, los N se refieren a la envoltura de tono que ya conocemos (véase p. 408):

ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,AR,DR,SR,RR,FAL,FDL

Los restantes parámetros aluden a la envoltura de volumen, estableciendo los volúmenes máximos y la velocidad de cambio de volumen en la ejecución de una nota requerida por la orden SOUND.

AR y DR (de −127 a 127); FAL y FDL (de 0 a 126)

AR (*Attack Rate*) establece la velocidad de subida de volumen. Aunque el software admite un valor negativo, en la práctica la escala va de 1 a 127. Indica el número de cambios de volumen por unidad de tiempo subiendo hasta alcanzar el FAL (*Final Attack Level*: máximo nivel de elevación), que da paso a la fase de apagamiento. La velocidad de apagamiento la controla DR (*Decay Rate*) de forma similar, pero con valores negativos, haciendo que el volumen decaiga hasta alcanzar el FDL (*Final Decay Level*: máximo nivel de apagamiento).

Aunque para los niveles máximos el software admite una escala de 0 a 126, el hardware normal sólo admite de 0 a 16, de modo que un valor FAL de 50 se adaptaría automáticamente a un volumen 6.

SR y RR (de −127 a 0)

La velocidad del sostenido (SR, *Sustain Rate*) y la velocidad del final (RR: *Release Rate*) aluden asimismo a cambios de volumen por unidad de tiempo. Ambos deben tomar valores negativos. El sostenido se prolonga hasta que se completa la duración establecida por la orden SOUND. Ello significa que si el tiempo de subida y el tiempo de apagamiento sumados son mayores o iguales que el tiempo de duración establecido, no habrá fase de sostenido aun siendo programada. El final comienza una

Ondas de luz

Los gráficos de Atari marcaron una senda que han seguido otros fabricantes

Los ordenadores personales Atari 400 y 800 son muy conocidos por sus sistemas de cartucho enchufable, pero las máquinas en sí mismas también poseen unas configuraciones para gráficos muy refinadas y disponibles en BASIC. Estas configuraciones, que son comunes a ambas máquinas, admiten nueve niveles de visualización en pantalla: tres modalidades para texto (ofreciendo diferentes tamaños de caracteres) y seis modalidades para gráficos. La resolución máxima que se puede obtener es de 320 × 192 puntos.

En los ordenadores Atari se puede escoger entre 16 colores, pero el número máximo que se puede visualizar simultáneamente es de cinco. Existen los juegos de caracteres ASCII estándar en mayúscula y minúscula, así como 37 caracteres para gráficos especiales de Atari. Estos caracteres se pueden utilizar en sentencias PRINT para construir visualizaciones y tablas en baja resolución. Los Atari también permiten controlar el movimiento del cursor desde un programa en BASIC. Esto se consigue utilizando los caracteres de control del cursor dentro de sentencias PRINT, para posicionar el texto que sigue en la pantalla a continuación. Los caracteres de control del cursor permiten mover el cursor arriba-abajo o atrás-adelante.

Una de las configuraciones más atrayentes de los Atari es su capacidad para emplear gráficos al estilo sprite, que se denominan gráficos PM (*Player-Missile*), que permiten que el usuario escriba juegos recreativos de acción rápida en BASIC. Sin embargo, no existen órdenes especiales en este lenguaje para utilizar los gráficos PM, y todo el trabajo necesario se ha de efectuar manipulando las posiciones de memoria de la RAM, mediante PEEK y POKE. Los gráficos PM serán analizados con mayor detalle en un próximo capítulo.

Modalidades de visualización

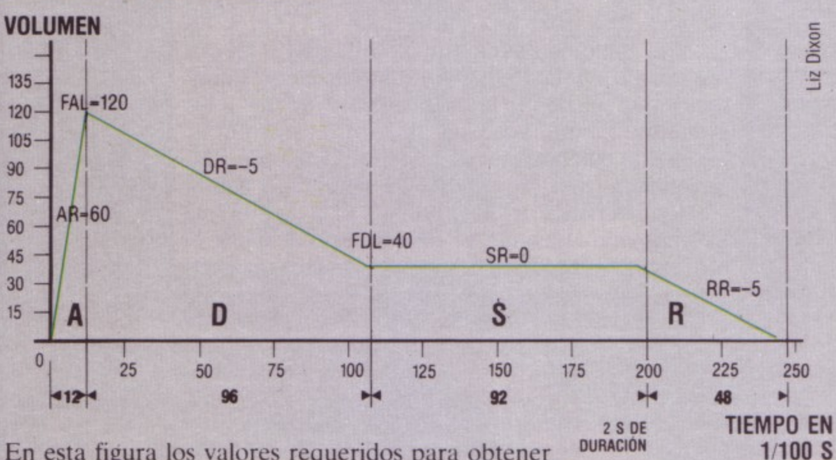
Las modalidades 0, 1 y 2 son para visualización de textos. Cuando se enciende la máquina, la visualización se establece en la modalidad 0 y a la pantalla se le da un formato de 24 filas, cada una de las cuales contiene 40 espacios de caracteres. En esta modalidad los caracteres de visualización se basan en el formato estándar ASCII de ocho por ocho. Los caracteres que se imprimen (PRINT) en la modalidad 1 son dos veces más anchos que los caracteres de la modalidad 0, pero conservan la misma altura; por su parte, los caracteres de la modalidad 2 son dos veces más altos y anchos que los de la modalidad 0.

A excepción de la modalidad 0, todas las modali-



vez completa la duración. El volumen baja hasta cero a la velocidad establecida, a menos que por el mismo oscilador no se dé comienzo a una nueva nota, lo que significa que el final se descarta, fenómeno evitable estableciendo "H" en "1" mediante una nueva orden SOUND &.

Envoltura de volumen



En esta figura los valores requeridos para obtener la envoltura similar a la de un piano serían los siguientes:

T = 6 AR = 60 SR = 0 FAL = 120
DR = −5 RR = −5 FDL = 40

duración SOUND = 40 (dos segundos)

Que expresaríamos así:

ENVELOPE 1,6,0,0,0,0,0,0,60,−5,0,−5,120,40

El siguiente programa emplea todas las órdenes de sonido en BASIC BBC para ejecutar una frase musical muy conocida con la envoltura de volumen propia del piano, y una breve envoltura de tono triangular repetida en el acorde final.

```
10 REM **COSMICO**
20 ENVELOPE 1,6,0,0,0,0,0,0,60,−5,0,−5,120,40
30 ENVELOPE 2,6,1,−1,1,1,2,1,60,−5,0,
   −5,120,40
40 FOR I = 1TO4: READ N
50 SOUND 1,1,N,20:REM **TOCAR LA SI SOL
   SOL**
60 SOUND &1001,0,0,5:NEXT I
70 SOUND &201,2,77,40: REM **ULTIMO**
80 SOUND &202,2,89,40: REM **RE MAYOR**
90 SOUND &203,2,109,40: REM **ACORDE**
100 DATA 137,145,129,85: REM **LA SI SOL
    SOL**
```

dades para gráficos poseen una pantalla dividida, reservándose las pocas líneas inferiores para datos diversos, como los mensajes de error. Para imprimir en el cuerpo principal de la pantalla en las modalidades 1 y 2, se ha de especificar un número de dispositivo. PRINT #6 permite imprimir texto en la

Mod.	Tipo	Filas	Cols.	Colores
0	texto	24	40	2
1	texto	20	20	5
2	texto	10	20	5
3	gráficos	20	40	4
4	gráficos	40	80	2
5	gráficos	40	80	4
6	gráficos	80	160	2
7	gráficos	80	160	4
8	gráficos	160	320	1

parte de gráficos de la pantalla. Las modalidades de la 3 a la 8 son para gráficos y permiten trazar en la pantalla puntos y líneas con grados variables de resolución y una selección de colores. La tabla que ofrecemos refleja la gama completa de las opciones que el usuario tiene a su disposición.

Órdenes en BASIC

El BASIC de Atari dispone de cierto número de órdenes para ayudar con los gráficos. Estas órdenes también funcionan, en forma modificada, en las tres modalidades para texto.

SETCOLOR a,b,c

Existen cinco registros de color para controlar su utilización en la pantalla, pero no todos ellos se emplean en todas las modalidades. SETCOLOR sirve para seleccionar los colores utilizados por estos cinco registros. En esta orden, a es el número del registro de color, 0-4; b es el número del color a utilizar, 0-15; y c permite que cada color se visualice en uno de los ocho niveles de brillo, escogiendo un número par entre 0 y 14.

COLOR n

Esta orden funciona de dos maneras, según se haya seleccionado una modalidad para texto o para gráficos. En las modalides 0, 1, y 2, n es un número de la escala entre 0 y 255. En su forma binaria, este número consta de ocho bits: los seis primeros aluden al código ASCII del carácter que se está trazando (PLOT), y los otros dos están reservados para la información de color relativa al carácter.

En las modalidades para gráficos, n asume un valor entre 0 y 3, y se utiliza para seleccionar un determinado registro de control de color cuando se traza (PLOT) un punto.

PLOT x,y

El origen de la pantalla Atari está situado en el rincón superior izquierdo de la pantalla. PLOT ilumina el punto de gráficos con las coordenadas (x,y). Del mismo modo, la orden POSITION:

POSITION x,y

coloca un cursor invisible en el punto (x,y) de la pantalla.

DRAWTO x,y

dibuja una línea recta (o lo más recta posible en las modalidades de resolución más baja) desde la antigua posición del cursor hasta (x,y). La línea:

X10 18, #6,0,0,"S:"

emplea la orden de Atari X10 para entrada-salida, que le permite al usuario rellenar o pintar una forma dibujada en la pantalla. Es bastante complicada, pero si se la emplea con cuidado puede dar buenos resultados. Una vez que se ha dibujado en la pantalla un área cerrada, se debe establecer el cursor en el rincón inferior izquierdo de la zona a colorear. La coloración comenzará desde la parte superior del área e irá rellenándola, entre los límites, hasta que la posición del cursor alcance la parte inferior. El color se establece mediante POKE 765,C, donde C es 1, 2 o 3, como en la orden COLOR.

Tamaño XL
Los gráficos de Atari son muy interesantes, pero no fáciles de utilizar, aunque tienen la ventaja de su amplia gama de modalidades para texto. El siguiente programa le demuestra al usuario la utilización de los caracteres de tamaño doble, junto con la orden POSITION, para imprimir en la pantalla un mensaje familiar:

```
10 REM *LETRAS
   GRANDES*
20 GRAPHICS 2+16
30 SETCOLOR 0,3,6
40 FOR X = 19TO8
   STEP−1
50 POSITION X,1
60 FOR J = 1 TO 100:
   NEXT J
70 PRINT #6; "CURSO "
80 NEXT X
90 FOR X = 19TO6
   STEP−1
100 POSITION X,3
110 FOR J = 1 TO 100:
    NEXT J
120 PRINT #6; "MI "
130 NEXT X
140 FOR X = 13TO7
    STEP−1
150 POSITION X,9
160 FOR J = 1 TO 100:
    NEXT J
170 PRINT #6;
    "COMPUTER "
180 NEXT X
190 SETCOLOR 0,5,5
200 FOR Y = 9TO5
    STEP−1
210 POSITION 7,Y
220 PRINT #6;
    "COMPUTER "
230 NEXT Y
240 GOTO 240
```

Observe que cuando se selecciona una modalidad, se puede contrarrestar el efecto de la pantalla dividida agregando 16 al número de modalidad

Jerga

Algunos de los términos que se utilizan en el mundo de la informática nacieron del modo más pintoresco

Muchas de las palabras que definen aspectos de la informática tienen un extraño origen. Toda profesión cuenta con un lenguaje especializado propio (palabras o frases que utilizan las personas que desempeñan esa actividad), pero ninguna ha creado tantos términos como la industria informática. Incluso se ha acuñado un vocablo para referirse a ellos: *buzzwords*.

Buzzwords

La palabra **BUZZWORD** se utilizó por primera vez a finales de los años sesenta, cuando a alguien del departamento de publicidad de la Honeywell le dio por crear un juego denominado *generador de "buzzwords"*. El juego se basa en tres columnas de diez palabras cada una, numeradas del 0 al 9. La primera columna es una lista de sustantivos, las dos restantes contienen adjetivos o modificadores indirectos que pueden ir gramaticalmente unidos a los primeros. Piense usted un número de tres cifras, busque las palabras correspondientes y se encontrará con una frase tan carente de significado como: "matriz heurística de diagnóstico". Si la dice en una conversación, imagine el desconcierto y hasta el asombro que causará.

Boot

BOOT es un apócope de *bootstrap*, como este vocablo lo es de *"to pull oneself up by one's bootstraps"* (pararse uno por sus propios pies). Un cargador de bootstrap es una rutina que se ejecuta automáticamente cada vez que a un ordenador se le proporciona alimentación eléctrica (el usuario de ordenadores especializado no se conforma con decir "encendido"). En aquellas máquinas que no poseen un sistema operativo en ROM, la rutina boot debe contener instrucciones para llamar a ese sistema operativo desde el disco.

Bit

Aunque la mayoría de los diccionarios afirman que se trata de una contracción de **BI**nary Digi**T** (dígito binario), parece igualmente probable que no sea más que una ampliación de su significado en inglés: brizna. Sin embargo, vale la pena tener presente que en el *argot* norteamericano un bit es asimismo la octava parte de un dólar y siempre se emplea de a pares: "dos bits", por ejemplo, equivalen a un cuarto (25 centavos).

Con frecuencia *bit* aparece como prefijo: como en *bit-slicing*, término que se utiliza para indicar que ciertos microprocesadores bastante sofisticados se pueden construir a partir de "bloques de construcción" de dos, cuatro u ocho bits, lo que da lugar a dispositivos de hasta 32 bits de capacidad. Igualmente, el sentido común informático nos dice que los programas que permanecen durante mucho tiempo sin utilizar originan errores adicionales e insolubles: este previsible fenómeno se conoce como *bit-decay*.

Turnkey

Para cuando una persona recibe un equipo informático, quizá por primera vez, la jerga ha reservado una nueva palabra. Muchas organizaciones comerciales trabajan con una firma de consultores de informática para instalar el hardware y el software, de modo que el cliente lo pueda recibir listo para funcionar. Esto se conoce como operación **TURNKEY,** porque todo lo que el cliente ha de hacer es sólo dar vuelta a la llave y empezar a servirse de la máquina.

Hardware

Software

HARDWARE y **SOFTWARE** son palabras bien conocidas (*hard* significa tangible y *soft*, lo contrario), pero existen asimismo otros dos tipos de *ware*: **FIRMWARE**, que significa que el software está encapsulado en el hardware (tal como sucede en la ROM o en la EPROM), y **LIVEWARE**, vocablo genérico ¡que engloba a todas aquellas personas que tienen la suerte de trabajar con ordenadores y de usarlos!

Basic

BASIC significa **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (código de instrucciones simbólico para principiantes). Al igual que sucede con muchos acrónimos, el término está tan extendido que uno llega a creer que primero vino la palabra y después la frase.

Baudio

La palabra **BAUDIO** (la velocidad a la cual se transmiten los datos) se escogió en honor de Emile Baudot, inventor de un código telegráfico que llegó a competir con el desarrollado por Samuel Morse.

Byte

BYTE es un vocablo informático que aparece con mucha frecuencia y, aunque sólo tiene 30 años, sus orígenes ya se han perdido entre las tinieblas. Hasta que apareció el microprocesador de ocho bits, los bits de un byte eran suficientes para codificar un único carácter, en algunas ocasiones seis, otras ocho. En aquel entonces, era muy raro que los ordenadores utilizaran una palabra de menos de 24 bits; y las máquinas diseñadas para aplicaciones específicas, llegaban hasta 64 bits. La evocación semántica de byte (mordisco) ha llevado a la acuñación del término **NYBBLE** (bocadito): ¡medio byte! Abusando todavía más de la analogía, un **GULP** (sorbo) es un pequeño grupo de bytes.

Bombas lógicas

Caballos de Troya

Los medios de comunicación se han apresurado siempre a hacer suyos los imaginativos términos de la jerga y en los últimos años se han dedicado a utilizar algunos de propio cuño. El tema de los delitos informáticos es un terreno especialmente fértil para el nacimiento de estos vocablos: **BOMBAS LÓGICAS** y **CABALLOS DE TROYA** son dos de los métodos supuestamente utilizados con fines fraudulentos. El primero da idea de unas instrucciones que se escriben en un programa de aplicaciones pero que permanecen inactivas (o sea, sin efecto alguno) hasta que el programa se ha ejecutado durante un tiempo suficiente para que el fraude (transferir dinero de una cuenta a otra, tal vez) no se pueda detectar. El término caballo de Troya alude a un programa disfrazado de otro para introducirse en el sistema.

Bomba de relojería

Una expresión parecida, pero que se refiere a una práctica legítima, es **BOMBA DE RELOJERÍA**. Alude a una técnica particularmente ingeniosa para proteger al software de gestión contra la piratería. Se trata de una codificación introducida dentro del propio paquete, la cual queda inhabilitada cuando el sistema lo instala un comerciante honrado. Sin embargo, en una copia pirata, la bomba entrará en funcionamiento al llegar una fecha determinada, probablemente en el momento en que la compañía dependa en gran medida del paquete. Al día siguiente de la "explosión" de la bomba, no sólo se habrán convertido en "basura" los archivos del usuario, sino que también se habrá destruido la copia del programa (a menos que el disco estuviera protegido contra una reescritura).

Basura

BASURA es una palabra que está presente en diversas frases del lenguaje de los usuarios de ordenadores. Por ejemplo, el acrónimo **GIGO** significa **G**arbage **In**, **G**arbage **O**ut (basura entra, basura sale), y éste es en realidad un recordatorio de que los ordenadores sólo procesan información y que, por lo tanto, uno no puede esperar resultados exactos si en primer lugar no se provee a la máquina de datos pertinentes.

ACUMULACIÓN DE INFORMACIÓN INSERVIBLE es la expresión con la que se designa el proceso interno que se podría muy bien utilizar en su ordenador personal si éste empleara una versión de BASIC que admitiera series dinámicas (es decir, series cuya longitud puede alterarse durante un programa). Cada vez que una serie aumenta en longitud, en la RAM se hace una nueva copia completa. De modo que si hubiera varias sentencias de la forma LET A$ = A$+"*" (en especial dentro de bucles), entonces al cabo de poco tiempo la memoria se llenaría por completo. En este punto, la ejecución del programa se interrumpirá temporalmente de forma automática y una rutina de la ROM denominada *recolector de información inservible* ordenará la zona de la serie y eliminará todas las secciones de series que hubieran quedado de un tratamiento anterior. Aunque el programa continuará cuando el recolector haya terminado su tarea, el proceso puede tardar segundos e incluso minutos, durante los cuales el ordenador no operará.

Apretón de manos

Un acuerdo comercial suele sellarse con un apretón de manos entre las partes interesadas; por analogía, en el lenguaje informático, un **APRETÓN DE MANOS** es el nombre que se le da a la señal electrónica que avisa que se ha completado un intercambio de datos.

0. Red	0. Integrada	0. De datos
1. Capacidad	1. Situacional	1. Interactiva
2. Base	2. Vertical	2. De sistemas
3. Expresión	3. Digitalizada	3. De diagnóstico
4. Palanca	4. Estocástica	4. Direccional
5. Matriz	5. Lineal	5. Aleatoria
6. Impresora	6. Heurística	6. Gráfica
7. Aplicación	7. Relativa	7. Alfanumérica
8. Jerarquía	8. Habitual	8. Esquemática
9. Imagen	9. Programable	9. Modular

Generador de "términos"
El término *buzzword* se utilizó por primera vez para describir un juego sencillo consistente en crear frases en jerga tecnológica sin ningún significado, pero muy "convincentes". Usted puede desarrollar su propio "generador de *buzzwords*" ideando tres columnas de diez palabras cada una, como nosotros hemos hecho aquí. La elección de un número al azar de tres dígitos "generará" un término muy erudito (a la violeta)

Commodore PET 4032

El Commodore PET fue el primer ordenador personal. Desde su introducción en el mercado, su hardware ha mejorado sensiblemente

En muchos sentidos, el Commodore PET (acrónimo de *Personal Electronic Transactor*) fue la máquina que inició el *boom* del microordenador. Cuando se lanzó al mercado, en 1977, estableció un estándar tan elevado que, en comparación, algunas máquinas más recientes casi pueden considerarse retrocesos. La carcasa metálica de la máquina original es buena muestra de su superioridad. Aparte de Memotech y de las máquinas para gestión más caras, las carcasas de la mayoría de los ordenadores recientes están moldeadas en plástico y van desde las aún aceptables hasta las decididamente malas. La fuente de alimentación eléctrica incorporada del PET es otro detalle que lo distingue claramente de muchos de sus competidores del mercado personal.



Chris Stevens

El teclado y el monitor del PET
El teclado de los primeros PET no era estándar; el de los más recientes se aproxima en mayor medida al estilo de los de las máquinas de escribir e incorpora los símbolos para gráficos en el frente de las teclas (exceptuando los modelos de gestión). Todos los PET poseen monitores incorporados: los últimos tienen pantalla de 12" (30 cm), con visualizaciones en verde sobre negro y una opción de columnas de 40 u 80 caracteres

Si bien al menos dos años antes de que se lanzara el PET al mercado ya existían máquinas de 8 bits e incluso de 16, éstas eran *kits* para montar o simples "sistemas mínimos" compuestos sólo por chips sobre un circuito impreso. El PET fue el primer microordenador en salir a la venta del que realmente se podía afirmar que para usarlo sólo había que enchufarlo. Las primeras versiones del PET tenían una grabadora en cinta incorporada, con control de motor, un monitor incorporado y BASIC en la ROM. Todo cuanto debía hacer un nuevo usuario para comenzar a trabajar con él era enchufarlo y encenderlo, y aparecía un mensaje tranquilizador:

```
COMMODORE BASIC VER. 1.0
7167 BYTES FREE
READY
```

El usuario podía entonces empezar a digitar y su trabajo podía ser almacenado en una cassette a toda prueba, sin necesidad de tener que enchufar diversos componentes entre sí ni de cargar de una cinta programas para el sistema de carga (o, peor aún, de tener que darle entrada mediante un teclado HEX, que era algo común en aquellos días).

A lo largo de su existencia, el BASIC de Commodore ha sido objeto de varias revisiones, y la última versión (4) se ha ampliado tanto como para convertirse en una nueva.

Otra configuración importante y exclusiva del PET es el juego de caracteres. Consta tanto del juego ASCII completo como de una gran variedad de gráficos de bloques; los usuarios de PET los han empleado de manera notablemente creativa, a pesar de la algo baja resolución de los caracteres. No obstante, un problema importante de la máquina era que los códigos generados por el teclado no se correspondían con el juego ASCII ni estaban acomodados a ningún orden estandarizado.

La intensa utilización de estos gráficos de bloques se ha reforzado merced a la existencia de una

Chip sincronizador
Cuando se enciende un ordenador, los circuitos tardan unos instantes en estabilizarse. Este sincronizador espera durante una fracción de segundo, al cabo de lo cual repone el microprocesador para que comience el intérprete de BASIC

Conexión dispositivos
Esta interface contiene un número de líneas útiles, incluyendo una puerta en paralelo de ocho bits y conexiones para interface de monitor externo. Es particularmente adecuada para conectar proyectos electrónicos de diseño casero

Puerta IEEE488
El PET fue el único de los primeros microordenadores que incluyó esta interface en paralelo. Debido a que podía direccionar hasta 15 periféricos, se utilizaba para activar tanto discos como impresoras. La IEEE488 también es la puerta estándar que se emplea para conectar en interface equipos científicos de laboratorio

6522
Este adaptador versátil para interfaces es parecido al 6520, pero contiene un registro para realizar desplazamientos para convertir datos de en paralelo a en serie y viceversa, así como dos sincronizadores programables que se pueden utilizar para controlar equipos externos

6520
Estos PIA (*Peripheral Interface Adaptors*: adaptadores de interfaces de periféricos) se encargan de la mayoría de las interfaces, incluyendo cassette y teclado

6502
El PET lo diseñó Chuck Peddle, de Commodore, de manera que no sorprende que se base en un microprocesador 6502, también diseñado por él. Aunque los ordenadores de gestión han optado por otros procesadores, el 6502 sigue gozando de popularidad entre los usuarios de ordenadores personales



Conexión cassette
Se ha de utilizar un aparato de cassette Commodore modificado especialmente. Cuando se introdujo por primera vez el PET, el aparato de cassette Commodore ofrecía mejor rendimiento que las unidades domésticas, pero ahora esa situación se ha invertido

Segunda conexión cassette
Los PET originales tenían un aparato de grabación incorporado. Ahora esta conexión se puede utilizar para agregar una segunda unidad y esto permite que los datos se puedan leer en una cinta, modificarlos, y luego escribir en otra

Conector para ampliación
Aquí están a disposición las señales de control de datos y de dirección, provenientes del microprocesador

ROM
El PET fue el primero en colocar en ROM el BASIC completo y el sistema operativo, iniciando una tendencia que han seguido casi todos los ordenadores personales

Altavoz piezoeléctrico
Los modelos posteriores fueron incorporando este dispositivo que se puede programar, por ejemplo, para producir un "gorjeo" cuando el usuario hace una entrada errónea

Conector para el teclado

Generador de caracteres
Además de los 64 caracteres alfanuméricos, el PET puede generar 64 símbolos para gráficos. El texto se puede visualizar en mayúsculas y minúsculas, a voluntad

RAM
Los PET poseen de 8 a 32 Kbytes como estándar. Mediante una modificación especial, esta cifra se puede ampliar a 96 Kbytes

Chris Stevens

COMMODORE PET

DIMENSIONES
480 × 440 × 300 mm

CPU
6502

VELOCIDAD DEL RELOJ
1 MHz

MEMORIA
32 Kbytes de RAM
20 Kbytes de ROM

VISUALIZACION EN VIDEO
25 líneas de 40 caracteres. Monitor de fósforo verde de 12" (30 cm) incorporado. 256 caracteres y símbolos para gráficos visualizables, o gráficos en baja resolución (50 × 80)

INTERFACES
IEEE488, puerta en paralelo de 8 bits para el usuario, cassette (2)

LENGUAJE SUMINISTRADO
BASIC, monitor de lenguaje máquina

OTROS LENGUAJES DISPONIBLES
PASCAL, COMAL, LISP

VIENE CON
Manual de instrucciones

TECLADO
Teclado estilo máquina de escribir; posee 64 teclas individuales con símbolos para gráficos inscritos en el frente. Un relleno de teclado numérico separado incluye teclas de función de calculadora

DOCUMENTACION
Commodore nunca ha gozado de mucho prestigio por la calidad de su documentación, aunque ésta ha mejorado mucho desde los primeros días

gama de impresoras que los reproducen sin necesidad de la complicada programación de bits de la cabeza de impresión. Por supuesto, esto significa que el número de impresoras aptas para emplearlos con el PET es limitado y la mayoría, aunque no todas, son de la propia casa Commodore.

Como consecuencia de estas características diferenciadoras, y a pesar de que existe una considerable cantidad de software para la máquina, es muy poco el que se ha traducido para otras máquinas. Igualmente, son pocos los programas de otras máquinas que se han convertido al BASIC del PET, porque la conversión por lo general implica un esfuerzo excesivo y resulta más sencillo simplemente volver a escribirlos. Por consiguiente, en cierto sentido la máquina se ha quedado un poco "aislada" en su pequeño mundo propio y las modificaciones que se producen en la industria en general apenas si inciden en ella.

Elementos subversivos

Con una cuidadosa planificación y un enfoque paso a paso se acorta el tiempo necesario para eliminar los errores de un programa

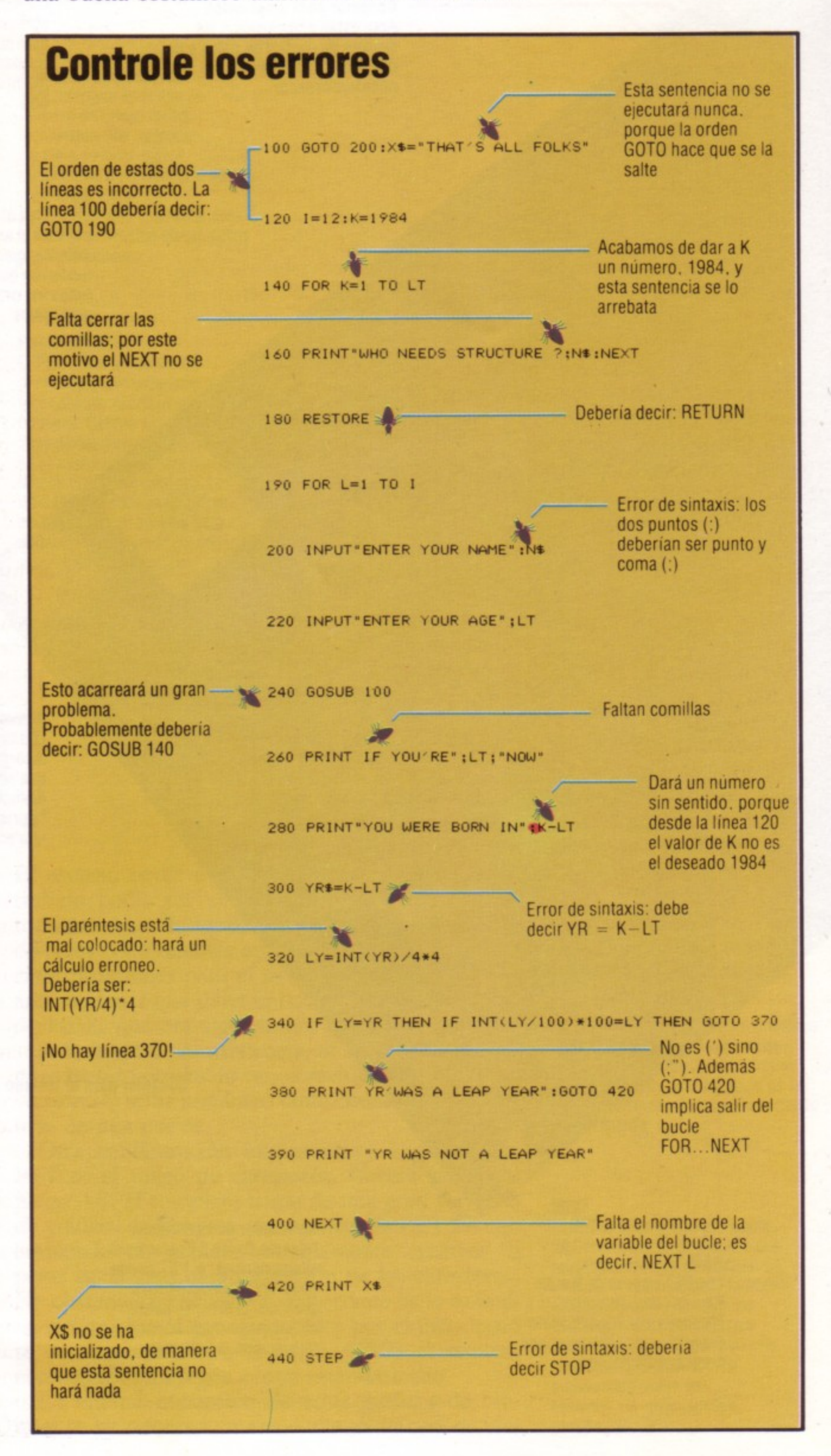
A medida que usted vaya adquiriendo mayor experiencia en la escritura de programas, también irá empeñándose cada vez más en "depurarlos". Los errores sintácticos y los errores de lógica, en los que incurren hasta los programadores de ordenadores más experimentados, se van haciendo cada vez menos frecuentes y menos problemáticos a medida que su experiencia aumenta. Vamos a dar algunas sugerencias para ayudarle a evitar los errores de programación y para aumentar su eficacia en la depuración del código.

Comencemos por donde un programa empieza: ¡por el cerebro de usted! Si desde el principio elabora equivocadamente el concepto de un programa, entonces lo más seguro es que al escribirlo quede plagado de errores.

Una idea mucho mejor es la de comenzar a escribir un programa enunciando primero el problema, a uno mismo o a otras personas, con la mayor claridad posible. Después, divida el problema en partes completas desde el punto de vista de la lógica (entrada, salida, algoritmos, estructuras de datos, procesos, etc.) y considere cada una de estas partes como un problema separado. De ser necesario, divida cada uno de estos problemas en subproblemas, y así sucesivamente, hasta que el problema original sea un conjunto estructurado de pequeños problemas, cada uno de los cuales le resultará sencillo de programar. Un enfoque formal, como utilizar un seudolenguaje o un diagrama de flujo, es esencial en la etapa de diseño para registrar (y preservar) la estructura del programa como un todo. Debe tratar de permanecer alejado del teclado hasta que honestamente considere que está en condiciones de saber cómo programar cada una de las partes del problema. Éste es el enfoque *top-down* de la programación, método que reduce notablemente el tiempo que se consume en la depuración.

Dividir los problemas en tareas resolubles le llevará a escribir programas que realmente sean conjuntos de subrutinas o procedimientos unidos por un programa principal esquematizado. Así resulta más fácil hallar los errores y usted puede construir una biblioteca de subrutinas a prueba de errores para utilizarlas en programas ulteriores. De otro modo, estaría continuamente inventando el fuego: cada vez que escribiera, por ejemplo, un programa para clasificar datos, volvería a resolver el problema de escribir una rutina de clasificación.

En la medida en que el BASIC se lo permita, trate en todo caso de utilizar nombres adecuados para las variables, aun cuando tenga que abreviarlos. NETO = BRUTO−IMPUESTO, por ejemplo, se explica por sí solo, y NT = BR−IM no es un mal sustituto; pero N = B−I es sumamente ambiguo y no le sugiere indicación alguna acerca de cuáles son las variables en cuestión. Es un buen hábito llevar una tabla de variables, que le indique todas las variables utilizadas en el programa y para qué sirven. Esto puede conducirle a normalizar la utilización de las variables (o sea, denominar ciertas variables siempre con la misma letra, p. ej., las contadoras de bucles) e impide que usted emplee la misma variable con distintas finalidades. Del mismo modo, es una buena costumbre almacenar los valores cons-



tantes en variables al principio del programa y remitirse a partir de entonces a estas variables. Esto hace que el programa sea más rápido y pulido, y el usuario puede cambiar estos valores sin tener que buscar por el programa cada vez que se producen.

Ni siquiera con este enfoque formal es fácil eliminar los errores. De modo que es muy importante adoptar un método disciplinario para hallarlos y erradicarlos. Los errores más comunes son los de tipo sintáctico, y por lo general el usuario los puede corregir según los vaya encontrando. Pero no siempre es éste el caso. Consideremos lo siguiente:

```
10 PRINT "**PULSE LA BARRA ESPACIADORA**"
20 PRINT "**CUANDO ESTE LISTO**"
```

Las líneas de esta clase suelen producir un mensaje de error cuando se ejecutan si no se las digita como dos líneas separadas. La línea 10 contiene 40 caracteres, de modo que al digitarla en una pantalla de 40 columnas, el cursor acaba donde empieza la segunda línea de la pantalla, con lo que fácilmente usted puede que se olvide de pulsar RETURN en la línea 10 antes de comenzar a digitar la línea 20. De ser así, entonces lo que en su programa parecerían ser dos líneas perfectas en realidad sería una sola línea con un error de sintaxis (el número 20) en la mitad. Una forma de captar estos errores consiste en listar todas las líneas sospechosas individualmente en vez de hacerlo como si fueran una parte del fragmento de un programa.

Los mensajes de error, cuando no son del todo incomprensibles, pueden inducir a confusión. Tomemos el ejemplo:

```
25 DATA 10.2,34,56.9,0.008,15.6
30 FOR K = 1 TO 5: READ N(K): NEXT K
```

La ejecución de estas líneas, al fallar, nos puede hacer pensar que hay un error de sintaxis en la línea 30, cuando en realidad el error está en la línea 25 (uno de los ceros se ha digitado equivocadamente con la letra O).

Los errores de codificación que no son de sintaxis son los más comunes, y por lo general son también los más difíciles de hallar. En este caso, resulta vital ser metódico. Empiece por tratar de descubrir aproximadamente en qué lugar del programa está el error. Esto es bastante fácil en los programas modulares bien estructurados y puede resultar aún más fácil mediante la orden TRACE, que hace que se imprima en la pantalla el número de línea en curso del programa mientras éste se ejecuta. Si su máquina no admite TRACE, entonces puede crear sentencias TRACE periódicamente a lo largo del programa (tal como PRINT "LINEA 150" al comienzo de la línea 150, p. ej.). Del mismo modo, puede usar la orden STOP para interrumpir la ejecución del programa en los lugares significativos del mismo, de manera que pueda examinar los valores de las variables cruciales. Puede hacerlo en la modalidad directa utilizando PRINT, o bien puede escribir una subrutina al final de su programa:

```
11000 REM IMPRIMIR LAS VARIABLES
11100 PRINT "MARCADOR,TAMAÑO,BANDERAS"
11200 PRINT MR;TM;B1;B2
11300 PRINT "MATRIZ TABLERO"
11400 FOR K = 1 TO 10: PRINT TB $(K): NEXT K
```

En consecuencia, cuando el programa llegue a una orden STOP usted puede digitar GOTO 11000 y obtener la visualización del estado actual de las variables. Incluso puede modificarlas (digitando, pongamos por caso, TM = 17 y pulsando RETURN), y restaurar el programa con la orden CONTinuar.



El primer "bug"

Para los programadores novatos los errores suelen asumir rasgos zoomórficos: se esconden del programador y burlan a placer todo esfuerzo por dar con ellos. En inglés se utiliza la palabra "bug" (bicho) para indicar que existe un error. El primer *bug* (al menos aquel del cual proviene el término) era realmente animado. Mientras intentaba eliminar el error de un programa que estaba desarrollando en 1945 para el Harvard Mrk II, Grace Hopper descubrió que una enorme polilla atrapada en el sistema electromecánico del ordenador era la causante del supuesto error

Tony Lodge

Una vez que haya descubierto que el error está escondido en ciertas líneas, o en una variable determinada, en ese momento ya estará cerca de poder eliminarlo, ¡pero vaya con cuidado! Pruebe los remedios uno a uno, para que pueda observar cuál es su efecto exacto en la ejecución. Es muy sencillo introducir diversas modificaciones entre una y otra ejecución, con la idea de librarse de un error y crear con ellas varios más, intentar corregir éstos ¡y, finalmente, olvidarse de cómo empezó todo!

Los bucles y las bifurcaciones, sobre todo cuando están "anidados", pueden ser terreno particularmente propicio para cometer errores y exigen especial atención tanto al escribirlos como al depurarlos. Consideremos este trozo de código:

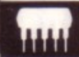
```
460 IF SM < 0 AND SC <> -1 THEN IF SC > 0 OR
    SM = SC-F9 THEN LT = 500
470 FOR C1 = 1 TO LT:FOR C2 = LT TO C1
    STEP -1
480 SC = SM + SC*C2
490 NEXT C2: SM = 0: NEXT C1
```

¿Y qué significa todo esto? Aun cuando usted supiera lo que se ha de hacer, ¿está seguro de conseguirlo? Colocar sentencias dentro de un bucle cuando éstas deberían estar fuera de él es una forma segura de facilitar la aparición de errores. Y lo mismo puede decirse cuando no se cubren todas las condiciones posibles al escribir sentencias IF... THEN. En este sentido, constituye un caso especial el que se produce cuando se escriben sentencias múltiples después de IF...THEN. Por ejemplo:

```
655 IF A$ = " " THEN GOTO 980:A$ = B$
660 PRINT A$
```

La sentencia A$ = B$ no se ejecutará jamás, porque o bien A$ = " ", en cuyo caso el control pasaría a la línea 980, o A$ <> " ", en cuyo caso se ignoraría el resto de la línea 655.

El mejor maestro para la depuración de errores es la experiencia, pero un enfoque paso a paso y un método disciplinado son ayudas inestimables. Tómeselo con tiempo y ¡NO SE ATEMORICE!



El show del láser

La tecnología del disco óptico (láser) hace accesibles al ordenador personal dos importantes aplicaciones: el video interactivo y el almacenamiento masivo de datos

La capacidad de almacenamiento interno del ordenador es importante, pero la capacidad de su sistema de almacenamiento masivo de datos a largo plazo resultará decisiva. Al cabo de un par de meses, el entusiasta usuario de un ordenador personal habrá acumulado una considerable cantidad de cassettes o varias cajas de discos. Como la mayoría de estos programas no se modificarán nunca, estarían mejor almacenados en cartuchos de ROM que en delicados medios magnéticos. Lo que sería muy útil es alguna forma de sistema de almacenamiento digital que fuera sólo de lectura, como un cartucho, pero que tuviera muchísima más capacidad.

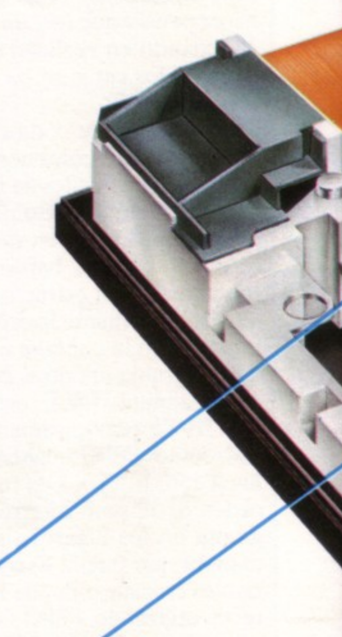
Dicho sistema existe y es el disco láser óptico. Lo corriente, sin embargo, es que este sistema se utilice en el hogar sólo como una alternativa a la grabadora de videocassette, para mostrar el material pregrabado. Otra utilización de la misma tecnología es el disco compacto de audio.

La diferencia entre estos dos tipos de sistemas (aparte de los diámetros de sus discos) estriba en sus métodos de operación. Mientras que un videodisco es un sistema analógico, un disco de audio compacto almacena su información en forma digital, es decir, como una secuencia de unos y ceros. Esta información se vuelve a convertir en la señal de audio original mediante un convertidor digital-analógico, que es el opuesto electrónico del proceso que en primer lugar creó la información. Dado que existen tantos campos eléctricos dispersos en el entorno doméstico, no es práctico utilizar los medios magnéticos como los discos flexibles para la grabación de video. En cualquier caso, la cantidad de información de un disco óptico puede ascender a millones de megabytes, y esto es mucho más de lo que puede retener hasta un disco Winchester.

Existen en el mercado varios sistemas de discos láser ópticos, pero hasta el momento el que ha obtenido mayor éxito es el que ha introducido Philips. Este sistema usa un disco plástico de 14" (35 cm) que en realidad no es sino una envoltura de protección. La información propiamente dicha está oculta dentro del plástico en forma de una serie de "agujeros" en una lámina metálica. Al igual que en un disco flexible, la información almacenada está catalogada en el videodisco, de manera que, con un tocadiscos adecuado, se puede pasar instantáneamente a cualquier unidad de información individual. Una vez que la cabeza de lectura está en la posición deseada, el rayo láser lee la información del disco. La luz pasa a través del plástico y cae sobre la superficie de la chapa metálica. Una célula fotosensible lee entonces la información a medida que los agujeros de la chapa van reflejando la luz. La información se graba en una única pista en espiral, a fotograma de video por revolución. Esto da un total de 54 000 fotogramas por cada cara del disco, o 36 minutos de ejecución.

Las principales aplicaciones potenciales de los discos ópticos en el campo de los ordenadores van por dos caminos. El primero, que ya existe en el mercado, es el del *video interactivo*. Un programa transmitido por televisión no es interactivo: el espectador no puede controlar el orden en que se le presentan las imágenes. Pero, con el video interactivo, la información visual y textual se almacena en un videodisco que está conectado con un ordenador. El disco se puede entonces utilizar como una biblioteca de referencia, con el texto visualizado superpuesto contra las imágenes de video en la pantalla de un televisor convencional. En respuesta a las indicaciones del ordenador, el usuario puede seleccionar "pistas" o "escenas" específicas del videodisco para que se reproduzcan. Por otro lado, el disco se puede emplear como medio de entrenamiento, visualizando en un televisor las acciones en movimiento o las imágenes fijas y dando entrada en el ordenador a las respuestas de quien se está entrenando a las preguntas importantes, pudiendo el ordenador controlar e informar sobre el rendimiento del usuario. Aún no están muy difundidas las interfaces entre un videodisco doméstico y un ordenador personal. Así y todo, Philips comercializa un modelo profesional de su *Laser Vision*, que puede hacer frente por sí mismo al video interactivo o se puede conectar en interface con un ordenador mediante una puerta IEEE488 o una RS232.

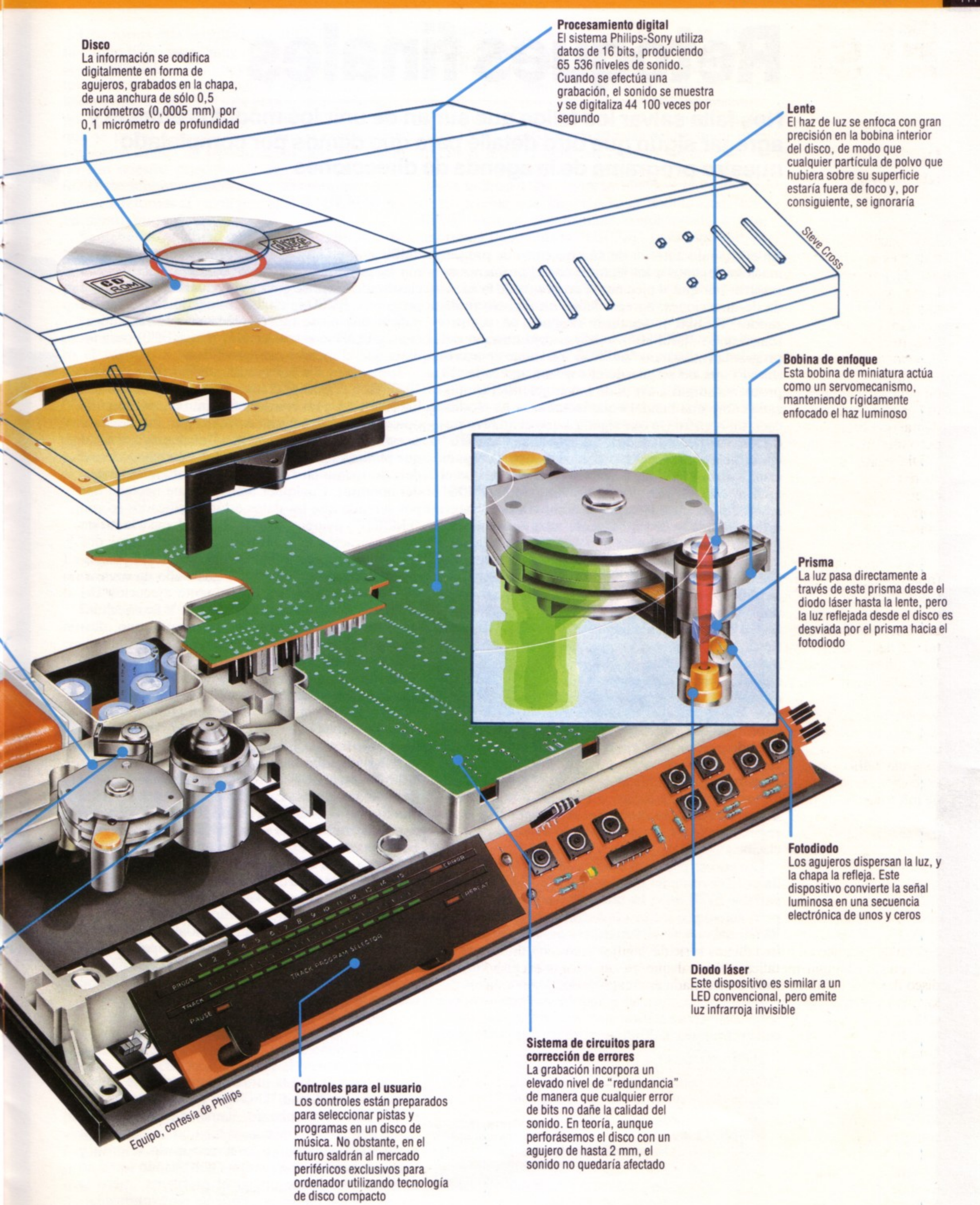
El otro campo en el que es probable que se explote la tecnología del disco óptico es en la provisión de software para ordenadores. Imagínese, por ejemplo, las ventajas de suministrar un ordenador con todo su software de sistemas (tratamiento de textos, base de datos, hoja electrónica y varias docenas de juegos) en un solo disco "incorruptible". Es probable que éste asuma el formato del disco compacto, pero hasta el momento no se ha equipado ningún tocadiscos compacto con una interface para ordenador. Con un mercado potencial tan enorme, es razonable esperar que en un plazo muy breve aparezcan los tocadiscos compactos domésticos con interfaces de este tipo, así como reproductoras exclusivas de discos compactos para ordenadores personales. Sony y Philips ya han hecho pública su intención de producir un tocadiscos exclusivo para ordenadores, denominado CDROM.



Motor lineal
El servomecanismo para mover el brazo de localización sobre el disco es simplemente una bobina que trabaja contra un resorte ligero. La disposición se parece mucho a la de un medidor de bobina móvil, tal como los medidores de corriente o de voltaje

Brazo de localización
El brazo gira centralmente sobre un pivote, está cuidadosamente equilibrado y es de giro libre. Por consiguiente, la cabeza de lectura describe un arco a través del disco

Motor
La velocidad de rotación del disco se controla con gran precisión utilizando un sistema de circuitos de realimentación. A medida que el brazo va del centro del disco al borde, la velocidad cambia de 500 a 200 r.p.m. para mantener constante la densidad de grabación


Disco
La información se codifica digitalmente en forma de agujeros, grabados en la chapa, de una anchura de sólo 0,5 micrómetros (0,0005 mm) por 0,1 micrómetro de profundidad

Procesamiento digital
El sistema Philips-Sony utiliza datos de 16 bits, produciendo 65 536 niveles de sonido. Cuando se efectúa una grabación, el sonido se muestra y se digitaliza 44 100 veces por segundo

Lente
El haz de luz se enfoca con gran precisión en la bobina interior del disco, de modo que cualquier partícula de polvo que hubiera sobre su superficie estaría fuera de foco y, por consiguiente, se ignoraría

Steve Cross

Bobina de enfoque
Esta bobina de miniatura actúa como un servomecanismo, manteniendo rígidamente enfocado el haz luminoso

Prisma
La luz pasa directamente a través de este prisma desde el diodo láser hasta la lente, pero la luz reflejada desde el disco es desviada por el prisma hacia el fotodiodo

Fotodiodo
Los agujeros dispersan la luz, y la chapa la refleja. Este dispositivo convierte la señal luminosa en una secuencia electrónica de unos y ceros

Diodo láser
Este dispositivo es similar a un LED convencional, pero emite luz infrarroja invisible

Sistema de circuitos para corrección de errores
La grabación incorpora un elevado nivel de "redundancia" de manera que cualquier error de bits no dañe la calidad del sonido. En teoría, aunque perforásemos el disco con un agujero de hasta 2 mm, el sonido no quedaría afectado

Controles para el usuario
Los controles están preparados para seleccionar pistas y programas en un disco de música. No obstante, en el futuro saldrán al mercado periféricos exclusivos para ordenador utilizando tecnología de disco compacto

TRACK PROGRAM SELECTOR
PROGR
TRACK
PAUSE
REPEAT

Equipo, cortesía de Philips

Retoques finales

Nos falta salvar los fallos que surjan de unir los módulos entre sí y agregar algún que otro detalle para que demos por completado nuestro programa de la agenda de direcciones

En el capítulo anterior de nuestro curso de programación, dejamos a los lectores con el problema de resolver por qué al ejecutar el programa de la agenda de direcciones, agregar luego un registro (utilizando *INCREG*), localizar después un registro (empleando *ENCREG*) y salir a continuación del programa (utilizando *SAPROG*) se tiene como resultado que no se guarde el registro agregado. El problema surgió a causa de la variable RMOD utilizada como una bandera que indica si se ha modificado un registro (lo que significa que el archivo pudiera estar desordenado). La subrutina *CLSREG* clasificaría el archivo en orden alfabético y después establecería RMOD en 0 en la suposición de que el archivo estuviera en orden. Ejecutando *SAPROG* se verificaba para ver que el archivo estuviera en orden (RMOD = 0) y no se molestaba en guardar el archivo si éste estaba en una situación clasificada.

Agregando un registro nuevo (mediante *INCREG*) establece RMOD en 1 (dado que se habría modificado un registro, es decir, se habría agregado un registro nuevo), pero *CLSREG* establecería RMOD en 0, indicando que el archivo se había clasificado. Sin embargo, lo que realmente se necesita, independientemente de que el archivo se haya clasificado o no, es una bandera para señalar la modificación de un registro y otra bandera aparte para indicar si el archivo está o no clasificado. Luego, las subrutinas que necesitan saber si el archivo está clasificado pueden verificar la bandera "clasificado", y las que necesiten saber si se ha modificado algún registro pueden verificar la bandera "modificado".

Los nombres adecuados para las dos banderas serían RMOD, para mostrar si se ha modificado un registro, y CLAR, para mostrar si el archivo se ha clasificado.

Cuando se presentó el programa en la p. 399, la línea 1230 contenía la sentencia LET SVED = 0. La variable SVED no se ha utilizado hasta el momento, pero cuando se incluyó la línea se comprendió que RMOD sola no sería suficiente. Se escogió para la variable el nombre SVED con la idea de que antes de que fuera necesario guardar algo (en cinta o disco) tendrían que cumplirse ciertas condiciones.

Un nombre más apropiado para esta bandera sería CLAR (para indicar que el archivo está clasificado). La línea 1230 original se ha cambiado por:

```
1230 LET CLAR = 1
```

Ahora hay cuatro estados posibles respecto a la situación del archivo de datos. Éstos son:

RMOD	CLAR	
0	0	No modificado, no clasificado (ilegal)
1	0	Modificado, no clasificado
0	1	No modificado, clasificado
1	1	Modificado, clasificado

RMOD = 0 y CLAR = 0 es ilegal porque el programa asegura que el archivo de datos siempre está clasificado antes de guardarlo. Cuando se ejecuta el programa, RMOD se establece en 0 (línea 1220) para indicar que no se ha efectuado ninguna modificación, y CLAR se establece en 1 (línea 1230) para indicar que el archivo está clasificado.

Toda operación que modifique un registro (como *INCREG*, *BORREG* o *MODREG*) establece RMOD en 1 y esta bandera no cambia con ninguna operación subsiguiente. A CLAR, que inicialmente se establece en 1, la restaura a 0 cualquier actividad que pudiera significar que los datos se hayan desordenado (como en *MODREG* si se altera el campo del nombre). Cualquier actividad que necesite dar por sentado que los datos están clasificados (como *ENCREG*) siempre verifica CLAR y llama a la rutina de clasificación si CLAR = 0. Utilizando estas dos banderas en lugar de RMOD únicamente, el programa puede concluir sin haber guardado un archivo de datos que durante la pertinente ejecución del programa no se hubiera modificado. Y no concluirá sin haberlo antes guardado, en caso de que, después de la modificación de un registro, se haya producido una clasificación.

La otra variable que hasta el momento no ha sido utilizada es CURS. Esta variable se emplea para guardar la posición "corriente" en la matriz de un registro después de que la rutina de búsqueda ha localizado uno. CURS no se restaura después de que se le ha asignado un valor; se la utiliza para llevar información acerca del registro objetivo a otras rutinas del programa. En las líneas 3320 y 3330 se ha modificado el final de la rutina *ENCREG* (búsqueda) para establecer el valor de CURS: o 0 si la búsqueda no da con el registro objetivo; y en MED si la búsqueda tuvo éxito.

La línea 13340 se bifurca a la subrutina *NINREG* si CURS es 0. Con ella lanzamos un mensaje para advertir que no se ha encontrado el registro y visualizamos la clave de búsqueda, NOMCAM$ (TAMA). *NINREG* nos devuelve el menú principal después de que se pulsa la barra espaciadora. *NINREG* se puede modificar con bastante facilidad para darle al usuario la oportunidad de:

```
PULSE RETURN PARA VOLVER A INTENTARLO O
BIEN BARRA ESPACIADORA PARA CONTINUAR
```

Podría parecer que la forma más sencilla de conseguir esto sería llamar *ENCREG* otra vez si se pulsara RETURN. Sin embargo, llamar a una subrutina desde dentro de sí misma, si bien no es "ilegal" en BASIC, sí "confunde" a la dirección de retorno y hará que la subrutina se repita aun cuando usted no lo desee. Se puede obviar el problema, ¡pero la programación empieza a ponerse muy intrincada!

Una forma más sencilla sería utilizar una bandera (como NREG para "ningún registro") y restaurarla en *WINREG*, permitiendo que la subrutina retorne de la manera normal, y forzando un salto hacia atrás, a *EJECUT*, en el programa principal; por ejemplo: 95 IF NREG = 0 THEN 80. Este enfoque se probó y resultó bien. Pero la codificación empezaba a tener un aspecto desaliñado. De acuerdo con nuestro principio de evitar las sentencias GOTO, decidimos mantener las cosas simples y limitarnos a retornar al menú principal si *ENCREG* no encuentra un registro.

Se debe hacer notar una pequeña adición a la línea 10490 en *MODNOM*. La variable numérica S también se debe restaurar (LET S = 0). No hacerlo puede, bajo ciertas condiciones inusuales, hacer que *MODNOM* funcione mal.

La otra rutina ejecutada en esta versión final del programa es *MODREG*. Esta rutina localiza primero el registro a modificar llamando a *ENCREG* (línea 14120). Esta línea llama a la línea 13030, no a la 13000, con el fin de suprimir la sentencia de *ENCREG* para limpiar la pantalla. Si no se puede localizar el registro, el programa retorna al menú principal de forma normal (en la línea 14130). Si se localiza el registro, se deja visualizado en la pantalla el registro objetivo y se les indica a los usuarios:

```
¿MODIFICAR NOMBRE?
PULSE RETURN PARA ENTRAR EL NUEVO NOMBRE
O BARRA ESPACIADORA PARA SIGUIENTE CAMPO
```

La rutina que averigua cuál de las dos opciones se requiere se puede hallar desde la línea 14190 hasta la 14280.

De la línea 14190 a la 14220 constituyen un bucle sencillo que sólo termina si se pulsa la barra espaciadora o RETURN. Si A$ no es CHR$(13) (el valor ASCII para un "retorno de carro") y no es un espacio (usted también podría emplear CHR$(32) en lugar de " "), I se restaurará y se repetirá el bucle. Si la tecla pulsada fue RETURN (es decir, que se ha de cambiar el campo del nombre), las líneas siguientes llenarán NOMCAM (CURS) con el nuevo nombre, establecerán RMOD, restaurarán CLAR, llamarán a *MODNOM* y llenarán MODCAM$(CURS) con el nombre estandarizado creado por *MODNOM* y localizado en MODCAM$(TAMA).

El resto de *MODNOM* funciona exactamente de la misma manera. Observe, no obstante, que modificar los otros campos establece RMOD pero no restaura a CLAR (véase línea 14490, p. ej.). La razón de ello es que sólo cambiar el campo del nombre implica que el archivo de datos puede estar desordenado, ya que está ordenado por nombre. Cambiar cualquier otro campo sólo indica que se ha cambiado un registro (RMOD = 1) y que cuando el programa termine se debe guardar el archivo.

La otra rutina ejecutada es *BORREG*, para eliminar un registro. Ésta es muy directa. Primero limpia la pantalla (línea 15020) y visualiza un mensaje explicando lo que está sucediendo. Luego llama a *ENCREG* para localizar el registro a eliminar. Entonces se ofrece una elección: pulsar RETURN para eliminar el registro o la BARRA ESPACIADORA para retornar al menú principal. También se visualiza un mensaje de aviso (línea 15160). Un enfoque aún mejor podría ser responder con un mensaje ¿ESTA SEGURO? si se pulsara RETURN y luego sólo borrar el registro si se pulsara la tecla S (es decir, IF INKEY$ = "S" THEN...).

BORREG no restaura la bandera CLAR. Dado que el archivo ya está en orden alfabético de nombres, borrar un registro completo no alteraría este orden. Sin embargo, sí significa que el archivo se ha modificado y, por consiguiente, RMOD se restaura en la línea 15340 y TAMA se reduce en uno en la línea 13550, para dejar constancia del hecho de que ahora el archivo tiene un registro válido menos. Todos los registros se desplazan "un lugar abajo" desde la línea 15260 hasta la 15320.

Es posible que también haya notado que *ENCREG* incluye una llamada condicionada a una subrutina denominada *LSTCUR* para imprimir el registro en curso localizado por *ENCREG*. Si usted no posee una impresora, simplemente reemplace la línea 13540 por una REM para futura ejecución y omita desde la línea 13600 a la 13690.

Con esto se completa el programa de la agenda de direcciones. Hemos tratado las opciones más importantes presentadas en el menú principal: hallar un registro, agregar un registro, modificar un registro, borrar un registro y salir del programa. La agenda de direcciones informatizada nos ha servido para ilustrar cómo un programador debe especificar, diseñar y ejecutar un programa. Una modificación esencial que habría de introducir quien tenga la intención de convertir el programa en software de aplicación, sería comprobar (y evitar) el problema que surgiría si TAMA fuera igual a 51. Esto sucedería nada más tener 50 registros en el archivo.

En el próximo capítulo analizaremos los estilos de programación y abordaremos algunos de los aspectos más avanzados de este lenguaje.

Complementos al BASIC



No disponen de esta orden el Commodore 64, Vic-20, BBC Micro y el Dragon 32

En el BBC Micro con una impresora en paralelo, inserte las siguientes líneas:

```
13605 VDU 2
13680 VDU 3
```

Éstas habilitan o inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670 sustituya PRINT por LPRINT. Para más información, consulte el manual del usuario.

En los Commodore, inserte las siguientes líneas:

```
13605  OPEN 4,4:CMD 4
13680  PRINT #4:CLOSE 4
```

Éstas habilitan e inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670, sustituya PRINT por LPRINT.

En el Dragon 32, inserte estas líneas:

```
13605  OPEN"0",-2
13680  CLOSE -2
```

Éstas habilitan e inhabilitan, respectivamente, la impresora. Desde la línea 13610 a la 13670, sustituya PRINT -2, (la coma forma parte de la orden) por LPRINT.

SPECTRUM

Para el Spectrum, el programa de la agenda de direcciones se publicará completo en el próximo capítulo del curso de programación BASIC.

Programa de la agenda de direcciones

```
10 REM 'PROGRAMA PRINCIPAL'
20 REM *INICIL*
30 GOSUB 1000
40 REM *PRESEN*
50 GOSUB 3000
60 REM *ELECCN*
70 GOSUB 3500
80 REM *EJECUT*
90 GOSUB 4000
100 IF OPCN<>9 THEN 60
110 END
1000 REM SUBRUTINA *INICIL*
1010 GOSUB 1100: REM SUBRUTINA *CREMAT*(CREAR MATRICES)
1020 GOSUB 1400: REM SUBRUTINA *LEARCH*(LEER ARCHIVOS)
1030 GOSUB 1600: REM SUBRUTINA *ESBAND*(ESTABLECER BANDERAS)
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SUBRUTINA *CREMAT*(CREAR MATRICES)
1110 DIM NOMCAM$(50)
1120 DIM MODCAM$(50)
1130 DIM CLLCAM$(50)
1140 DIM CIUCAM$(50)
1150 DIM PROCAM$(50)
1160 DIM TELCAM$(50)
1170 DIM INDCAM$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAMA = 0
1220 LET RMOD = 0
1230 LET CLAR = 1
1240 LET CURS = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SUBRUTINA *LEARCH*
1410 OPEN "I",#1,"DAT.AGCO"
1420 INPUT #1,TEST$
1430 IF TEST$ = "@VACIO" THEN GOTO 1530: REM CERRAR Y RETORNAR
1440 LET NOMCAM$(1) = TEST$
1450 INPUT #1,MODCAM$(1),CLLCAM$(1),CIUCAM$(1),PROCAM$(1),TELCAM$(1)
1460 INPUT #1,INDCAM$(1)
1470 LET TAMA = 2
1480 FOR L = 2 TO 50
1490 INPUT #1,NOMCAM$(L),MODCAM$(L),CLLCAM$(L),CIUCAM$(L),
     PROCAM$(L)
1500 INPUT #1,TELCAM$(L),INDCAM$(L)
1510 LET TAMA = TAMA + 1
1520 IF EOF(1) = -1 THEN LET L = 50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SUBRUTINA *ESBAND*
1610 REM ESTABLECE BANDERAS DESPUES DE *LEARCH*
1620 REM
1630 REM
1640 IF TEST$ = "@VACIO" THEN LET TAMA = 1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SUBRUTINA *PRESEN*
3010 PRINT CHR$(12): REM LIMPIAR PANTALLA
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(11);"*BIEN VENIDO A LA*"
3070 PRINT TAB(9);"*AGENDA COMPUTERIZADA*"
3080 PRINT TAB(12);"*DE MI COMPUTER*"
3090 PRINT
3100 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
3120 IF INKEY$<> " " THEN L = 0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SUBRUTINA *ELECCN*
3510 REM
3520 IF TEST$ = "@VACIO" THEN GOSUB 3860: REM SUBRUTINA *PRIMERA*
3530 IF TEST$ = "@VACIO" THEN RETURN
3540 REM 'IMMENU'
3550 PRINT CHR$(12)
3560 PRINT "SELECCIONE UNO DE LOS SIGUIENTES"
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT "1. HALLAR REGISTRO (DE NOMBRE)"
3610 PRINT "2. HALLAR NOMBRES (DE NOMBRE INCOMPLETO)"
3620 PRINT "3. HALLAR REGISTRO (DE CIUDAD)"
3630 PRINT "4. HALLAR REGISTRO (DE INICIAL)"
3640 PRINT "5. LISTAR TODOS LOS REGISTROS"
3650 PRINT "6. AGREGAR REGISTRO NUEVO"
3660 PRINT "7. MODIFICAR REGISTRO"
3670 PRINT "8. BORRAR REGISTRO"
3680 PRINT "9. SALIR Y GUARDAR"
3690 PRINT
3700 PRINT
3710 REM 'ASOPCN'
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 1 TO 1
3760 PRINT "DE ENTRADA A OPCION (1 - 9)"
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = "" THEN I = 0
3800 NEXT I
3810 LET OPCN<1 THEN L = 0
3810 LET OPCN = VAL(A$)
3820 IF OPCN < 1 THEN L = 0
3830 IF OPCN > 9 THEN L = 0
3840 NEXT L
3850 RETURN
3860 REM SUBRUTINA *PRIMERA*(VISUALIZAR MENSAJE)
3870 LET OPCN = 6
3880 PRINT CHR$(12): REM LIMPIAR PANTALLA
3890 PRINT
3900 PRINT TAB(10);"NO HAY REGISTROS EN"
3910 PRINT TAB(7);"EL ARCHIVO. DEBERA EMPEZAR"
3920 PRINT TAB(8);"POR AGREGAR UN REGISTRO"
3930 PRINT
3940 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
3960 IF INKEY$ <> " " THEN B = 0
3970 NEXT B
3980 PRINT CHR$(12): REM LIMPIAR PANTALLA
3990 RETURN
4000 REM SUBRUTINA *EJECUT*
4010 REM
4020 REM
4030 REM
4040 IF OPCN = 1 THEN GOSUB 13000: REM *INCREG*
4050 REM 2 ES *ENCNOM*
4060 REM 3 ES *ENCCIU*
4070 REM 4 ES *ENCINI*
4080 REM 5 ES *LISREG*
4090 IF OPCN = 6 THEN GOSUB 10000: REM *INCREG*
4100 IF OPCN = 7 THEN GOSUB 14000: REM *MODREG*
4110 IF OPCN = 8 THEN GOSUB 15000: REM *BORREG*
4120 IF OPCN = 9 THEN GOSUB 11000: REM *SAPROG*
4130 REM
4140 RETURN
10000 REM SUBRUTINA *INCREG*
10010 PRINT CHR$(12): REM LIMPIAR PANTALLA
10020 INPUT "DE ENTRADA AL NOMBRE";NOMCAM$(TAMA)
10030 INPUT "DE ENTRADA A LA CALLE";CLLCAM$(TAMA)
10040 INPUT "DE ENTRADA A LA CIUDAD";CIUCAM$(TAMA)
10050 INPUT "DE ENTRADA A LA PROVINCIA";PROCAM$(TAMA)
10060 INPUT "DE ENTRADA AL NUMERO DE TELEFONO";TELCAM$(TAMA)
10070 LET RMOD=1: LET CLAR=0: REM MODIFICADO Y NO CLASIFICADO
10080 LET INDCAM$(TAMA) = STR$(TAMA)
10090 LET TEST$ = ""
10100 GOSUB 10200: REM *MODNOM*
10110 LET OPCN = 0
10120 LET TAMA = TAMA + 1
10130 REM
10140 REM
10150 RETURN
10200 REM RUTINA *MODNOM*
10210 REM CONVIERTE CONTENIDO DE NOMCAM$ A MAYUSCULAS,
10220 REM ELIMINA CARACTERES EXTRAÑOS Y ALMACENA EN EL ORDEN:
10230 REM APELLIDO+ESPACIO+NOMBRE DE PILA EN MODCAM$
10240 REM
10250 LET N$ = NOMCAM$(TAMA)
10260 FOR L = 1 TO LEN(N$)
10270 LET TEMP$ = MID$(N$,L,1)
10280 LET T = ASC(TEMP$)
10290 IF T > = 97 THEN T = T - 32
10300 LET TEMP$ = CHR$(T)
10310 LET P$ = P$ + TEMP$
10320 NEXT L
10330 LET N$ = P$
10340 REM LOCALIZAR ULTIMO ESPACIO
10350 FOR L = 1 TO LEN(N$)
10360 IF MID$(N$,L,1) = " " THEN S = L
10370 NEXT L
10380 REM ELIMINAR CARACTERES EXTRAÑOS Y ALMACENAR
10390 REM NOMBRE DE PILA EN CNOM$
10400 FOR L = 1 TO S - 1
10410 IF ASC(MID$(N$,L,1)) > 64 THEN CNOM$ = CNOM$+MID$(N$,L,1)
10420 NEXT L
10430 REM ELIMINAR CARACTERES EXTRAÑOS Y ALMACENAR
10440 REM APELLIDO EN SNOM$
10450 FOR L = S + 1 TO LEN(N$)
10460 IF ASC(MID$(N$,L,1)) > 64 THEN SNOM$ = SNOM$ + MID$
      (N$,L,1)
10470 NEXT L
10480 LET MODCAM$(TAMA) = SNOM$ + " " + CNOM$
10490 LET P$ = " ": LET N$ = " ": LET SNOM$ = " ": LET
      CNOM$ = " ": LET S = 0
10500 RETURN
11000 REM SUBRUTINA *SAPROG*
11010 REM CLASIFICA Y GUARDA ARCHIVO
11020 REM SI ALGUN REGISTRO HA SIDO
11030 REM MODIFICADO (RMOD = 1)
11040 REM O NO HA SIDO CLASIFICADO (CLAR = 0)
11050 REM RMOD = 0 Y CLAR = 0 ES ILEGAL
11060 REM
11070 IF RMOD = 0 AND CLAR = 1 THEN RETURN
11080 IF RMOD = 1 AND CLAR = 0 THEN GOSUB 11200: REM *CLSREG*
11090 GOSUB 12000: REM *GRDREG*
11100 RETURN
11200 REM SUBRUTINA *CLSREG*
11210 REM CLASIFICA TODOS LOS REGISTROS EN ORDEN ALFABETICO
11220 REM MEDIANTE MODCAM$ Y ACTUALIZA INDCAM$
11230 REM
11240 REM
11250 LET S = 0
11260 FOR L = 1 TO TAMA - 2
11270 IF MODCAM$(L)>MODCAM$(L + 1) THEN GOSUB 11350
11280 NEXT L
11290 IF S = 1 THEN 11250
11300 REM
11310 REM
11320 LET CLAR = 1: REM ESTABLECE BANDERA 'ARCHIVO CLASIFICADO'
11330 REM
11340 RETURN
11350 REM SUBRUTINA *INTERCALAR*
11360 LET TNOMC$ = NOMCAM$(L)
11370 LET TMODC$ = MODCAM$(L)
11380 LET TCLLC$ = CLLCAM$(L)
```

```
11390 LET TCIUC$ = CIUCAM$(L)
11400 LET TPROC$ = PROCAM$(L)
11410 LET TTELC$ = TELCAM$(L)
11420 REM
11430 LET NOMCAM$(L) = NOMCAM$(L + 1)
11440 LET MODCAM$(L) = MODCAM$(L + 1)
11450 LET CLLCAM$(L) = CLLCAM$(L + 1)
11460 LET CIUCAM$(L) = CIUCAM$(L + 1)
11470 LET PROCAM$(L) = PROCAM$(L + 1)
11480 LET TELCAM$(L) = TELCAM$(L + 1)
11490 LET INDCAM$(L) = STR$(L)
11500 REM
11510 LET NOMCAM$(L + 1) = TNOMC$
11520 LET MODCAM$(L + 1) = TMODC$
11530 LET CLLCAM$(L + 1) = TCLLC$
11540 LET CIUCAM$(L + 1) = TCIUC$
11550 LET PROCAM$(L + 1) = TPROC$
11560 LET TELCAM$(L + 1) = TTELC$
11570 LET INDCAM$(L + 1) = STR$(L + 1)
11580 LET S = 1
11590 REM
11600 RETURN
12000 REM SUBRUTINA *GRDREG*
12010 REM
12020 REM
12030 OPEN "O",#1,"DAT.AGCD"
12040 REM
12050 FOR L = 1 TO TAMA - 1
12060 PRINT #1,NOMCAM$(L);",";MODCAM$(L);",";CLLCAM$(L);",";CIUCAM$(L)
12070 PRINT #1,PROCAM$(L);",";TELCAM$(L);","INDCAM$(L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
13000 REM SUBRUTINA *ENCREG* (HALLAR REGISTRO)
13010 PRINT CHR$(12): REM LIMPIAR PANTALLA
13020 REM
13030 IF CLAR = 0 THEN GOSUB 11200: REM *CLSREG*
13040 PRINT
13050 PRINT
13060 PRINT TAB(10);"BUSCANDO UN REGISTRO"
13070 PRINT TAB(13);"POR EL NOMBRE"
13080 PRINT
13090 PRINT TAB(7);"DIGITE EL NOMBRE COMPLETO"
13100 PRINT TAB(3);"POR ORDEN NOMBRE DE PILA APELLIDO"
13110 PRINT
13120 PRINT
13130 REM
13140 INPUT "EL NOMBRE ES ";NOMCAM$(TAMA)
13150 GOSUB 10200: REM SUBRUTINA *MODNOM*
13160 LET SCHKEY$ = MODCAM$(TAMA)
13170 REM
13180 REM
13190 REM
13200 REM
13210 REM
13220 LET INF = 1
13230 LET SUP = TAMA - 1
13240 FOR L = 1 TO 1
13250 LET MED = INT((INF+SUP)/2)
13260 IF MODCAM$(MED) <> BUSCLV$ THEN L = 0
13270 IF MODCAM$(MED) < BUSCLV$ THEN INF = MED + 1
13280 IF MODCAM$(MED) > BUSCLV$ THEN SUP = MED - 1
13290 IF INF > SUP THEN L = 1
13300 NEXT L
13310 REM
13320 IF INF > SUP THEN LET CURS = 0
13330 IF INF <= SUP THEN LET CURS = MED
13340 IF CURS = 0 THEN GOSUB 13700: REM *NINREG*
13350 IF CURS = 0 THEN RETURN
13360 REM
13370 REM
13380 PRINT CHR$(12)
13390 PRINT
13400 PRINT TAB(11);"*HALLADO REGISTRO*"
13410 PRINT
13420 PRINT "NOMBRE:",NOMCAM$(CURS)
13430 PRINT "CALLE:",CLLCAM$(CURS)
13440 PRINT "CIUDAD:",CIUCAM$(CURS)
13450 PRINT "PROVINCIA:",PROCAM$(CURS)
13460 PRINT "TELEFONO:",TELCAM$(CURS)
13470 PRINT
13480 PRINT TAB(2);"PULSE CUALQUIER LETRA PARA IMPRIMIR"
13490 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
13500 FOR I = 1 TO 1
13510 LET A$ = INKEY$
13520 IF A$ = "" THEN I = 0
13530 NEXT I
13540 IF A$<> " " THEN GOSUP 13600: REM *LSTCUR*
13550 RETURN
13600 REM SUBRUTINA *LSTCUR* (LISTAR REGISTRO EN CURSO)
13610 LPRINT
13620 LPRINT "NOMBRE:",NOMCAM$(CURS)
13630 LPRINT "CALLE:",CLLCAM$(CURS)
13640 LPRINT "CIUDAD:",CIUCAM$(CURS)
13650 LPRINT "PROVINCIA:",PROCAM$(CURS)
13660 LPRINT "TELEFONO:",TELCAM$(CURS)
13670 LPRINT
13680 LPRINT
13690 RETURN
13700 REM SUBRUTINA *NINREG* (NO HALLADO REGISTRO)
13710 PRINT CHR$(12): REM LIMPIAR PANTALLA
13720 PRINT TAB(9);"*NO HALLADO REGISTRO*"
13730 PRINT TAB(4);"*EN LA FORMA: ";NOMCAM$(TAMA);" *"
13740 PRINT
13750 PRINT TAB(0);"(PULSE BARRA ESPACIADORA PARA CONTINUAR)"
13760 FOR I = 1 TO 1
13770 IF INKEY$<>" " THEN I = 0
13780 NEXT I
13790 RETURN
14000 REM SUBRUTINA *MODREG* (MODIFICAR REGISTRO)
14010 REM
14020 PRINT CHR$(12): REM LIMPIAR PANTALLA
14030 PRINT
14040 PRINT
14050 PRINT
14060 PRINT
14070 PRINT TAB(6);"*PARA MODIFICAR UN REGISTRO*"
14080 PRINT TAB(1);"*LOCALICE PRIMERO EL REGISTRO DESEADO*"
14090 REM
14100 REM
14110 REM
14120 GOSUB 13030: REM SUBRUTINA *ENCREG* SIN CLS
14130 IF CURS = 0 THEN RETURN: REM NO HALLADO REGISTRO
14140 PRINT
14150 PRINT TAB(11);"¿MODIFICAR NOMBRE?"
14160 PRINT
14170 PRINT TAB(0);"PULSE RETURN PARA ENTRAR EL NUEVO NOMBRE"
14180 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14190 FOR I = 1 TO 1
14200 LET A$ = INKEY$
14210 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
14220 NEXT I
14230 IF A$ = CHR$(13) THEN INPUT "NUEVO NOMBRE";NOMCAM$(CURS)
14240 IF A$ = CHR$(13) THEN RMOD = 1
14250 IF A$ = CHR$(13) THEN CLAR = 0
14260 IF A$ = CHR$(13) THEN NOMCAM$(TAMA) = NOMCAM$(CURS)
14270 IF A$ = CHR$(13) THEN GOSUB 10200: REM SUBRUTINA *MODNOM*
14280 IF A$ = CHR$(13) THEN LET MODCAM$(CURS) = MODCAM$(TAMA)
14290 PRINT
14300 PRINT TAB(11);"¿MODIFICAR CALLE?"
14310 PRINT
14320 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA CALLE"
14330 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14340 FOR I = 1 TO 1
14350 LET A$ = INKEY$
14360 IF A$<> CHR$(13) AND A$<> " " THEN I = 0
14370 NEXT I
14380 IF A$ = CHR$(13) THEN RMOD = 1
14390 IF A$ = CHR$(13) THEN INPUT "NUEVA CALLE";CLLCAM$(CURS)
14400 PRINT
14410 PRINT TAB(11);"¿MODIFICAR CIUDAD?"
14420 PRINT
14430 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA CIUDAD"
14440 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14450 FOR I = 1 TO 1
14460 LET A$ = INKEY$
14470 IF A$<> CHR$(13) AND A$<>" " THEN I = 0
14480 NEXT I
14490 IF A$ = CHR$(13) THEN RMOD = 1
14500 IF A$ = CHR$(13) THEN INPUT "NUEVA CIUDAD";CIUCAM$(CURS)
14510 PRINT
14520 PRINT TAB(9);"¿MODIFICAR PROVINCIA?"
14530 PRINT
14540 PRINT TAB(0);"PULSE RETURN PARA ENTRAR LA NUEVA PROVIN"
14550 PRINT TAB(0);"O BARRA ESPACIADORA PARA SIGUIENTE CAMPO"
14560 FOR I = 1 TO 1
14570 LET A$ = INKEY$
14580 IF A$<> CHR$(13) AND A$<>" " THEN I = 0
14590 NEXT I
14600 IF A$ = CHR$(13) THEN RMOD = 1
14610 IF A$ = CHR$(13) THEN INPUT "NUEVA PROVINCIA";PROCAM$
      (CURS)
14620 PRINT
14630 PRINT TAB(5);"¿MODIFICAR NUMERO DE TELEFONO?"
14640 PRINT
14650 PRINT TAB(0);"PULSE RETURN PARA ENTRAR NUEVO TELEFONO"
14660 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
14670
14680 LET A$ = INKEY$
14690 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
14700 NEXT I
14710 IF A$ = CHR$(13) THEN RMOD = 1
14720 IF A$ = CHR$(13) THEN INPUT "NUEVO TELEF";TELCAM$ (CURS)
14730 REM
14740 REM
14750 RETURN
15000 REM SUBRUTINA *BORREG* (BORRAR REGISTRO)
15010 REM
15020 PRINT CHR$(12): REM LIMPIAR PANTALLA
15030 PRINT
15040 PRINT
15050 PRINT
15060 PRINT
15070 PRINT TAB(7);"*PARA BORRAR UN REGISTRO*"
15080 PRINT TAB(1);"*LOCALICE PRIMERO EL REGISTRO DESEADO*"
15090 REM
15100 REM
15110 REM
15120 GOSUB 13030: REM SUBRUTINA *ENCREG* SIN CLS
15130 IF CURS = 0 THEN RETURN: REM NO HALLADO REGISTRO
15140 PRINT
15150 PRINT TAB(6);"¿DESEA BORRAR ESTE REGISTRO?"
15160 PRINT TAB(1);"*AVISO*-- NO HAY SEGUNDA OPORTUNIDAD"
15170 PRINT
15180 PRINT TAB(8);"PULSE RETURN PARA BORRAR"
15190 PRINT TAB(3);"O BARRA ESPACIADORA PARA CONTINUAR"
15200 FOR I = 1 TO 1
15210 LET A$ = INKEY$
15220 IF A$<>CHR$(13) AND A$<>" " THEN I = 0
15230 NEXT I
15240 IF A$ = " " THEN RETURN
15250 FOR L = CURS TO TAMA - 2
15260 LET NOMCAM$(L) = NOMCAM$(L + 1)
15270 LET MODCAM$(L) = MODCAM$(L + 1)
15280 LET CLLCAM$(L) = CLLCAM$(L + 1)
15290 LET CIUCAM$(L) = CIUCAM$(L + 1)
15300 LET PROCAM$(L) = PROCAM$(L + 1)
15310 LET TELCAM$(L) = TELCAM$(L + 1)
15320 LET INDCAM$(L) = STR$(L)
15330 NEXT L
15340 LET RMOD = 1
15350 LET TAMA = TAMA - 1
15360 REM
15370 REM
15380 REM
15390 RETURN
```

Grace Hopper

Grace Hopper fue una gran impulsora de los lenguajes de alto nivel, y a ella se debe la identificación del primer "bug"


Cortesía de Sperry Ltd.

Suele pensarse que la ciencia de la informática por lo general es un club "sólo para hombres". Pero las mujeres van ocupando cada vez más posiciones junto a los hombres, en pie de igualdad, en el desarrollo y la aplicación de los ordenadores. Una de las pioneras de la informática fue Grace Hopper, cuya aportación más trascendente revolucionó el campo del software: fue la autora del primer compilador y colaboró de manera destacada en la elaboración y puesta a punto del lenguaje COBOL. También fue la primera persona que aisló un *bug* (véase p. 433) en un ordenador y logró "depurarlo".

Después de hacer un trabajo de posgrado en Yale, Grace Hopper regresó a su universidad de origen, Vassar, como miembro de la facultad de matemáticas. Allí permaneció hasta la edad de 39 años, cuando fue llamada para el servicio de guerra en el Naval Ordinance Computation Project. En 1945 se le ordenó que fuera a la Universidad de Harvard para ayudar al físico Howard Aiken en la construcción de un ordenador. En 1937 Aiken había propuesto a la IBM la idea de construir un ordenador utilizando equipos de tabulación adaptados. Su primer ordenador, aunque de diseño mecánico, tuvo el éxito suficiente como para animar a IBM a invertir en un modelo mejorado que pudiera emplear relés electromecánicos. Así se construyó una máquina que se denominó Harvard Mark II.

En aquellos primeros días las máquinas habían de programarse volviendo a tender sus cables para cada nueva tarea. De modo que, en el caluroso verano de 1945, Grace Hopper se encontró literalmente atrapada entre los cables del ordenador. Las necesidades de la guerra exigían con toda urgencia el procesamiento de datos balísticos, y el comandante Aiken solía irrumpir en la sala para preguntarle: "¿Por qué no está usted haciendo números, Hopper?" Una avería del ordenador lo impedía. Cuando finalmente se descubrió que el fallo se debía a una enorme polilla que había penetrado por las ventanas abiertas y que había quedado atrapada en el interruptor de un relé, Grace le replicó sucintamente: "¡Estamos desinsectando la máquina!" Este primer *bug* del que se tiene constancia se extrajo con sumo cuidado del relé con un par de pinzas y se conserva en el Museo Naval de Virginia en el cuaderno de bitácora del Harvard Mark II. Está pegado con cola junto a la entrada de las 15.45 del 9 de septiembre de 1945.

Aquel mismo año dos ingenieros, John Mauchly y Presper Eckert, estaban construyendo otro ordenador, el ENIAC (véase p. 46). Finalizada la guerra, los dos hombres crearon su propia empresa para fabricar una versión comercial de la máquina, e invitaron a Grace a unirse a su equipo. La ayuda más valiosa que aportó al desarrollo de este ordenador, denominado UNIVAC (UNIVersal ACcounting Machine) la constituyó la creación de software para el mismo. Y fue mientras intentaba construir programas de aplicaciones empresariales en el UNIVAC cuando Grace buscó por primera vez un atajo para ahorrarse la necesidad de volver a escribir ciertas subrutinas que se repetían una y otra vez. Valiéndose de lo que entonces se tuvo como brillante estratagema, la de que un ordenador podía escribir sus *propios* programas, Grace creó el primer lenguaje de programación, junto con el compilador necesario para traducirlo a código de lenguaje máquina. Se le dio el nombre de "A-0". Cuando este compilador fue presentado en público por primera vez suscitó la incredulidad entre los profesionales de la informática, quienes pensaban que las máquinas sólo podían realizar operaciones aritméticas o manipular símbolos. Se quedaron atónitos viendo cómo un ordenador saltaba a una subrutina de su biblioteca al encontrarse con un verbo en imperativo al comienzo de lo que era casi una sencilla locución en inglés.

En mayo de 1959 la capitana Hopper fue invitada al Pentágono para formar parte de una comisión de trabajo que estaba intentando crear y estandarizar un lenguaje para ordenadores de uso comercial. En menos de un año la comisión dio a la luz la primera versión del COmmon Business Oriented Language (COBOL). Grace colaboró valiosamente en los esfuerzos de la comisión por aunar los mejores logros de cada uno de los lenguajes existentes y, por consiguiente, crear un lenguaje óptimo para las empresas en virtud de su calidad. Prueba del éxito que consiguió la comisión es que el COBOL continúa siendo hoy en día uno de los lenguajes más ampliamente utilizados.

COBOL

El COBOL fue uno de los primeros lenguajes de programación que se escribieron con la intención de hacerlos fácilmente accesibles a los no matemáticos. El lenguaje favorece la utilización de procedimientos generalizados escritos en un estilo narrativo del inglés, en vez de rutinas codificadas sólo válidas para un problema determinado.
Un programa en COBOL consta de cuatro unidades. El nombre del programa, su autor y otra información de referencia constituyen la división de identificación (*Id-division*). Aunque se supone que los programas en COBOL se pueden ejecutar en diferentes máquinas, todos los detalles relativos al ordenador para el cual se escribió originalmente el programa se incluyen en la división Entorno (*Environment division*). Y dado que en muchas partes del mismo programa se podrían emplear los mismos datos, el COBOL posee una división de Datos (*Data division*) aparte. Por último, los procedimientos que manipularán los datos se codifican en la división Procedimientos (*Procedure division*)